Editores:

José María de Fuentes - Lorena González - Jose Carlos Sancho - Ana Ayerbe - María Luisa Escalante

Investigación en Ciberseguridad
Actas de las VII Jornadas Nacionales
(JNIC 2022)

27-29 de junio de 2022

Palacio Euskalduna, Bilbao

Editores:

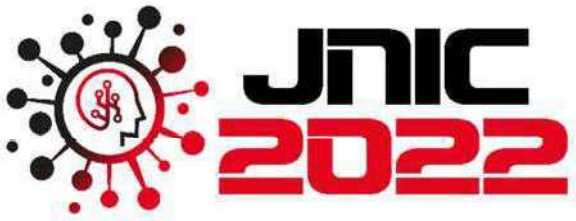
José María de Fuentes

Lorena González

Jose Carlos Sancho

Ana Ayerbe

María Luisa Escalante



I.S.B.N : 978-84-88734-13-6

QuantumSolver: Librería para el desarrollo cuántico

José Daniel Escánez-Expósito
Universidad de La Laguna
Tenerife, Spain
jdanielescanez@gmail.com

Pino Caballero-Gil
Universidad de La Laguna
Tenerife, Spain
pcaballe@ull.edu.es

Francisco Martín-Fernández
IBM Research
NY, USA
paco@ibm.com

***Resumen*—En este documento se presenta una breve descripción de la propuesta en desarrollo de un *toolset* cuántico *opensource*, llamado *QuantumSolver*, con licencia MIT. La herramienta desarrollada incluye varios algoritmos con distintas funcionalidades, como la generación de números aleatorios, la resolución del problema de Bernstein-Vazirani y la distribución de claves cuánticas (protocolo BB84). Se exponen aquí los principales detalles de la implementación del *toolset*, así como algunas conclusiones obtenidas de la investigación realizada de las funcionalidades incluidas.**

***Index Terms*—Computación cuántica, Qiskit, Números aleatorios, Algoritmo de Bernstein-Vazirani, Protocolo BB84**

Tipo de contribución: *Investigación en desarrollo*

I. Introducción

El interés en las tecnologías relacionadas con la computación cuántica ha crecido notablemente en los últimos años, cobrando el tema un gran auge hoy en día. Su utilidad para vulnerar los algoritmos criptográficos actuales ha generado la ya imperiosa necesidad de la creación de nuevos protocolos seguros para las comunicaciones. Por otra parte, dadas algunas de sus curiosas propiedades, está claro que el fomento de este modelo de computación generará importantes avances tecnológicos para el conjunto de la sociedad [1].

El principal objetivo de este trabajo es fomentar el uso de tecnologías cuánticas mediante el desarrollo de un *toolset opensource* de algoritmos cuánticos, con repositorio público en GitHub y licencia MIT [2], persiguiendo la abstracción y el encapsulamiento sencillos de *software* cuántico con diferentes funcionalidades. Entre las librerías que han implementado algunos de los algoritmos y protocolos que se tratan aquí destacan las publicadas en [3] y [4].

QuantumSolver está dirigido tanto a un usuario totalmente ajeno a la informática, que, por ejemplo, desee obtener un número aleatorio gracias a la computación cuántica; como a un programador experimentado, que, por ejemplo, aspire a contribuir en la implementación de esta librería. Debido a que la propuesta pretende satisfacer a una amplia variedad de posible público de los programas ejecutables disponibles, se ha desarrollado la posibilidad de ejecución del software por medio de dos interfaces diferentes: Interfaz por Línea de Comandos (*Command Line Interface*, CLI) e Interfaz Web. Esta última está más orientada al público general, y permite que cualquier usuario pueda ejecutar algoritmos cuánticos en *hardware* cuántico real, sin tener ningún tipo de experiencia previa con la programación informática.

II. Detalles de la implementación

QuantumSolver es una librería cuántica desarrollada en Python3 gracias a *Qiskit*, que es el SDK de código abierto que IBM ofrece para trabajar con ordenadores cuánticos a nivel de pulsos, circuitos y módulos de aplicación [5]. Cuenta con dos componentes principales: *QExecute* y *QAlgorithmManager*.

II-A. QExecute

QExecute es el motor de ejecución de *QuantumSolver*. Se encarga de la autenticación contra los servicios de IBM, que ofrecen acceso a su *hardware* (tanto *hardware* cuántico real como simuladores) por medio de un *API token* de *"IBM Quantum Experience"* [6] [7]. Además cuenta con un modo invitado para no generar la inevitable necesidad al usuario de obtener el *token* teniendo que crear una cuenta en *IBM Quantum*. En este modo solo se permite ejecutar haciendo uso del simulador local *'aer_simulator'*, por lo que no se podrá utilizar el *hardware* cuántico real proporcionado por IBM. *QExecute* cuenta con métodos para la visualización del listado de los *backends* disponibles y la selección del deseado para realizar la ejecución. Además, es el componente encargado de realizar la propia ejecución de los circuitos cuánticos.

II-B. QAlgorithmManager

QAlgorithmManager es el gestor de algoritmos cuánticos de *QuantumSolver*. Se encarga de agrupar y listar todos los algoritmos disponibles, además de seleccionar el que se desee ejecutar. También permite gestionar los argumentos de los diferentes algoritmos y del intercambio de información entre ellos y el programa principal.

II-C. QAlgorithm

QAlgorithm es la entidad que se corresponde con un algoritmo cuántico cualquiera. Se trata de una clase abstracta que puede servir como plantilla para añadir de manera intuitiva un nuevo algoritmo a la librería. Cualquier entidad válida derivada de esta representa un algoritmo que *QuantumSolver* puede ejecutar. Estas entidades, siguiendo la plantilla de *QAlgorithm*, contienen información relevante sobre el algoritmo en cuestión: nombre, descripción, parámetros, maneras en que se debe analizar y tratar el resultado de la ejecución del circuito, y en que se deben interpretar y comprobar los parámetros que sean introducidos como una lista de cadenas de texto. El método principal de la entidad es la generación parametrizada del circuito cuántico correspondiente al algoritmo.

III. Programa principal

En la pantalla de inicio, el programa principal de *QuantumSolver* ofrece las alternativas de, o bien ejecutar el modo

invitado, o bien autenticarse usando un *API token* de IBM. En cualquier caso, se despliega un menú que contiene las opciones de visualización y selección de los *backends* y algoritmos disponibles. Una vez elegido un algoritmo, se solicita la introducción de los parámetros ligados a él. Cuando *backend*, algoritmo y parámetros han sido establecidos, se despliegan dos opciones. Por una parte, se permite ejecutar el algoritmo una única vez y obtener el resultado, además de una representación gráfica del circuito. Por otra parte, es posible ejecutar el algoritmo varias veces para observar su comportamiento representado en un histograma generado. A esta última opción se la ha denominado modo experimental.

IV. Interfaces

Para *QuantumSolver* se ha desarrollado una versión web que cuenta con un *backend* en Python3, utilizando el *framework* Flask, y un *frontend* usando TypeScript, React, HTML5 y CSS. De las dos interfaces ofrecidas para ejecutar *QuantumSolver*, la interfaz web (ver Fig. 1) es más intuitiva para el público general que la basada en línea de comandos (ver Fig. 2), reuniendo ambas las mismas funcionalidades.

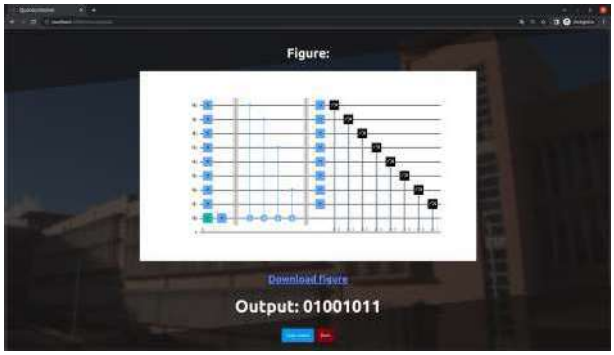


Figura 1. Interfaz web de *QuantumSolver*

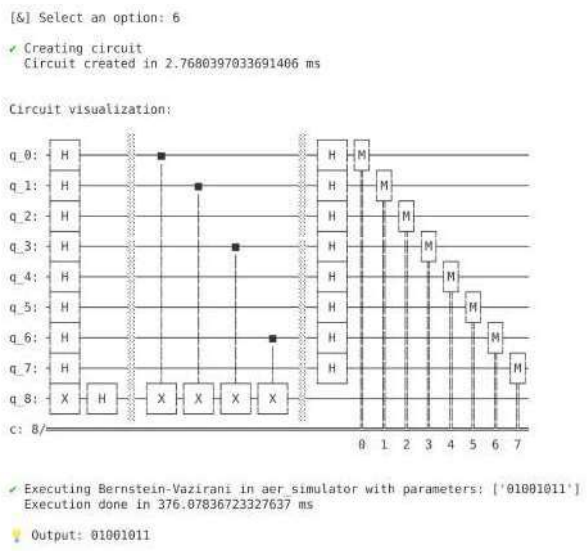


Figura 2. Interfaz *CLI* de *QuantumSolver*

V. Generación de números aleatorios

El algoritmo cuántico *QRand* implementado en *QuantumSolver* recibe como parámetro un número natural (n) y permite generar un circuito de cuya ejecución resulta un número aleatorio entre 0 y $2^n - 1$. Su funcionamiento se basa en la inicialización de n cúbits, por defecto a $|0\rangle$; la aplicación de una puerta lógica Hadamard [8] [9] a cada uno, para generar un estado de superposición en el que se tenga la misma probabilidad de medir 0 o 1 (ver Ec. (1)); y finalmente la medición del resultado, haciendo colapsar cada cúbit en un estado aleatorio e interpretándose como un número binario.

$$|00\ldots0\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \tag{1}$$

VI. Algoritmo de Bernstein-Vazirani

El algoritmo cuántico de Bernstein-Vazirani [10] incluido en *QuantumSolver* recibe como parámetro una clave formada por una cadena binaria de tamaño n, que se utiliza para codificar un oráculo que brinda información acerca de dicha clave. Concretamente, ante una cadena candidata, la información que devuelve el oráculo es la confirmación de si el número de coincidencias de 1 entre las cadenas clave y candidata es par o impar (ver Ec. (2)).

$$f_s(x) = (s * x)(mod\ 2) \tag{2}$$

La Ec. (2) refleja que el oráculo realiza el producto binario entre las parejas de bits de la clave a adivinar y la cadena candidata, y luego le aplica una puerta XOR o suma módulo 2 a los n bits resultantes. Clásicamente, se podría resolver este problema con n consultas al oráculo.

$$\begin{cases} f_s(100\ldots0) = s_0 \\ f_s(010\ldots0) = s_1 \\ f_s(001\ldots0) = s_2 \\ \ldots \\ f_s(000\ldots1) = s_{n-1} \end{cases} \tag{3}$$

En el caso cuántico se necesita una única consulta al oráculo, que devolverá correctamente la clave con un 100 % de probabilidad (sin contar con posibles errores de ruido generados por el *hardware*).

La implementación realizada del algoritmo [11] requiere $n+1$ cúbits. De ellos, n son para codificar la entrada (formada por n cúbits con valor $|0\rangle$). El restante es adicional, para la salida del oráculo cuántico (inicializado con el valor $|1\rangle$, obtenido al aplicar una puerta cuántica X [9] a un cúbit que por defecto tiene el valor $|0\rangle$). Además, se deberán aplicar puertas lógicas Hadamard a los $n+1$ cúbits antes y después del oráculo; excepto a la salida del mismo, que no lo precisará dado que no será medida. Internamente, este oráculo debe implementarse aplicando puertas *CNOT* con control en aquellos cúbits que se correspondan con los bits de la clave que estén a 1, y objetivo en la salida del oráculo. Esta puerta *CNOT* es el "equivalente" al *XOR* clásico.

La caracterización del oráculo, aplicado sobre la cadena x candidata, se muestra en la Ec. (4).

$$|x\rangle \xrightarrow{f_s} (-1)^{s\cdot x}|x\rangle \tag{4}$$

La transformación realizada de los n cúbits que codifican la entrada $|00\ldots0\rangle$ al aplicarles las puertas Hadamard de la inicialización se muestra en la Ec. (5).

$$|00\ldots0\rangle \xrightarrow{H^{\otimes n}} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \tag{5}$$

La aplicación del oráculo cuántico se muestra en la Ec. (6).

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} |x\rangle \xrightarrow{f_a} \frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{a\cdot x}|x\rangle \tag{6}$$

El paso final, aplicando a cada cúbit anterior una puerta Hadamard, se muestra en la Ec. (7).

$$\frac{1}{\sqrt{2^n}} \sum_{x\in\{0,1\}^n} (-1)^{a\cdot x}|x\rangle \xrightarrow{H^{\otimes n}} |a\rangle \tag{7}$$

Se observa que, al aplicar las operaciones descritas, el resultado es la clave codificada en el oráculo.

VII. Protocolo BB84

El protocolo criptográfico BB84 [12] para la distribución de claves cuánticas forma parte de la librería *QuantumSolver*.

VII-A. Entidades

Se ha implementado una entidad principal (*Participant*) y sus entidades derivadas (*Sender* y *Receiver*). Se supone que una instancia de la clase *Sender* quiere comunicarse con otra de la clase *Receiver*, de forma que la comunicación sea secreta gracias al protocolo BB84, que permite la generación de una libreta de un solo uso que solo comparten emisor y receptor. La simulación del canal cuántico se describe mediante circuitos cuánticos de Qiskit [13]. La única diferencia entre las entidades *Sender* y *Receiver* es que la primera tiene un método para enviar un mensaje (inicializando un circuito cuántico) y la segunda para recibirlo (añadiendo la fase de medición al circuito). La clase base *Participant* contiene los métodos para la generación y muestra de valores, ejes, claves y libretas de un solo uso, entre otros.

VII-B. Fundamento

La entidad emisora escoge al azar valores para codificar cada uno de los cúbits a transmitir y ejes en los que codificarlos, resultando las posibilidades indicadas en la Tabla I.

Tabla I
Posibilidades valor - eje y circuitos asociados

Valor	Eje	Circuito
$\|0\rangle$	Z	$q:$ ——
$\|0\rangle$	X	$q:$ —H—
$\|1\rangle$	Z	$q:$ —X—
$\|1\rangle$	X	$q:$ —X—H—

A continuación, la entidad receptora recibe esos circuitos (cada uno representando un cúbit) y, de manera aleatoria, elige ejes en los que medirlos. Aproximadamente el 50 % de los cúbits serán medidos correctamente, es decir, en el mismo eje que el emisor. El restante 50 % deberá ser descartado dado que al medirlos en el eje equivocado, se tendrá exactamente un 50 % de probabilidad de emitir el valor codificado correcto, lo que implica la pérdida de la correspondiente información. Para saber cuáles son los valores a descartar, ambas entidades hacen públicos los ejes en los que se midieron los cúbits, dado que no hay riesgo al realizar tal acción. Así desechan aquellos valores en los que los ejes no coinciden.

Los valores resultantes tras los descartes podrían ser considerados la clave generada, pero antes hay que verificar su seguridad. Se da una incoherencia entre la clave enviada por el emisor y la recibida por el receptor medida con el eje correcto, en aproximadamente el 50 % de los cúbits que hayan sido medidos en el eje incorrecto por un atacante [14]. En la Tabla II se ilustran los 4 casos posibles de mediciones de cúbits, todos con un 25 % de probabilidad de ocurrir.

En el último caso de la Tabla II se observa que, con una probabilidad del 50 %, se aborta el protocolo por detectar la comunicación comprometida. Teniendo en cuenta que cada uno de los cuatro casos tiene una probabilidad del 25 %, cuando se envía un único cúbit el protocolo es abortado con una probabilidad $p_{detectar}$ del 12.5 %. Cuando se envían n cúbits, dado que cuando se detecta algún ataque el protocolo se aborta, la probabilidad de abortarlo se deduce de la de no detectar ningún ataque, a partir de la Ec. (8).

$$1-(1-p_{detectar})^n = 1-\left(1-\frac{1}{4}*\frac{1}{2}\right)^n = 1-\left(\frac{7}{8}\right)^n \tag{8}$$

Este valor también se puede obtener aplicando el Teorema de Bayes con las probabilidades de no detectar ataque sobre cúbit no desechado (3/4), y sobre cúbit desechado (1).

Para verificar el resultado del protocolo, el receptor publica la mitad de la clave obtenida pues así el emisor puede compararla con la correspondiente mitad de su clave y, si ambas coinciden, la ejecución del protocolo se considera segura y se puede utilizar la mitad restante de la clave generada como clave secreta compartida. En caso contrario, se deduce que alguna entidad intermedia ha lanzado un ataque de escucha secreta o *eavesdropping*, interceptando los cúbits enviados y midiéndolos antes de remitirlos al receptor legítimo.

VII-C. Programa

La librería *QuantumSolver* permite la ejecución de una implementación realizada del protocolo BB84. Al ejecutar el programa, se despliega un menú que facilita la visualización y selección el *backend* disponible. Tras elegirlo, se dan dos opciones al usuario. Por una parte, puede correr el algoritmo una única vez y visualizar la traza entre los diferentes participantes en la comunicación, para lo que es necesario especificar una cadena de caracteres como mensaje y un valor entre 0 y 1 como densidad de intercepción (la probabilidad con la que el receptor intermedio medirá cada cúbit). Por otra parte, puede ejecutarlo varias veces y mostrar un mapa de calor que representa con colores más claros las condiciones en que ha habido más ocasiones en las que la comunicación ha sido considerada segura, y en colores más oscuros las comunicaciones en las que se detectado una intercepción. Para este modo experimental, se deben especificar diferentes condiciones: la longitud máxima del mensaje en número de bits, que definirá el eje x en el mapa de calor, este tomará

Tabla II
CASOS POSIBLES DE LAS MEDICIONES DE UN CÚBIT EN LA FASE DE VERIFICACIÓN DE BB84

Medición de emisor y receptor legítimos	Medición del atacante intermedio	Conclusión sobre ese cúbit
Distinto eje	Eje del emisor	Es desechado en la fase de descarte de los valores, aunque el receptor intermedio lo haya medido con un 100 % de probabilidad de obtener valor correcto
Distinto eje	Eje del receptor	Es desechado en la fase de descarte de los valores, aunque el receptor intermedio lo haya medido con un 50 % de probabilidad de obtener valor correcto (total incertidumbre del valor)
Mismo eje	Mismo eje que ambos	Ha sido interceptado por el atacante sin que se aborte el protocolo, con un 100 % de probabilidad de que el valor final sea correcto
Mismo eje	Eje contrario a ambos	Hay un 50 % de probabilidad de abortar el protocolo, en caso de que en el emisor colapse con el valor contrario al emitido por el emisor

valores enteros positivos hasta ese máximo; el valor del *step* de la densidad de intercepción, que definirá el eje y del mapa de calor, este tomará $\frac{1}{step}$ valores entre 0 y 1; y el número de repeticiones para cada instancia generada del problema.

Las Fig. 3 y 4 muestran dos ejemplos de mapas de calor generados, correspondientes a los parámetros de la Tabla III.

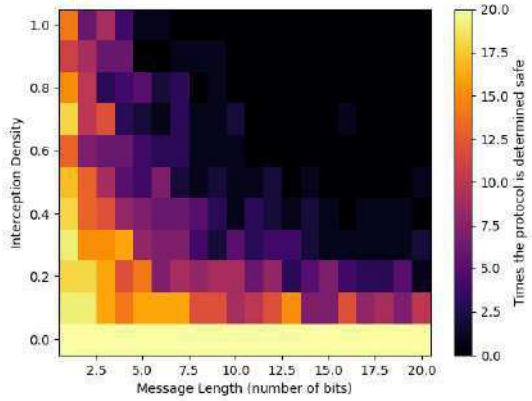


Figura 3. Primer ejemplo de mapa de calor de BB84

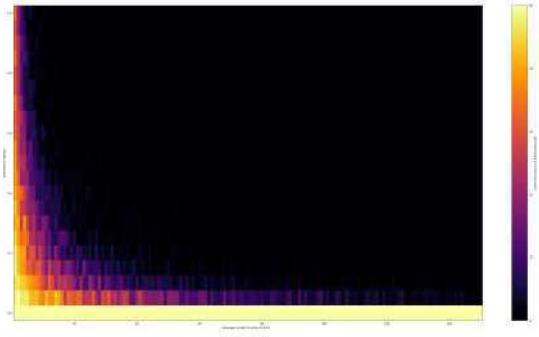


Figura 4. Segundo ejemplo de mapa de calor de BB84

Tabla III
PARÁMETROS RELEVANTES DE LOS MAPAS DE CALOR

Figura	Longitud máxima del mensaje	*Step* de densidad de intercepción	Repeticiones de cada instancia	Tiempo de ejecución (aprox.)
3	20 bits	0.1	20	5 minutos
4	150 bits	0.05	50	16 horas

VIII. CONCLUSIONES

La implementación presentada de la librería para el desarrollo cuántico *QuantumSolver* permite, de manera sumamente accesible, la ejecución de diversos algoritmos cuánticos en *hardware* cuántico real y simuladores proporcionados por IBM. También ofrece una sencilla arquitectura de entidades, respaldada por una batería de pruebas unitarias (*unitary test*), que facilita enormemente la adición de nuevos algoritmos a la librería. Precisamente, el objetivo principal en cuanto a la continuación del desarrollo de la propuesta es ampliar el número de algoritmos ahora disponibles en el *toolset*, incluyendo posiblemente los algoritmos B92, E91 y de Grover.

AGRADECIMIENTOS

Esta investigación ha sido posible gracias al proyecto RTI2018-097263-B-I00 financiado por el Ministerio de Ciencia, Innovación y Universidades, la Agencia Estatal de Investigación y el Fondo Europeo de Desarrollo Regional, y a la Cátedra de Ciberseguridad Binter-Universidad de La Laguna.

REFERENCIAS

[1] E. Gibney, “Hello quantum world! Google publishes landmark quantum supremacy claim”, Nature, vol. 574, no. 7779, pp. 461-462, 2019.
[2] J. D. Escánez, “QuantumSolver”. [Online]. Available: https://github.com/alu0101238944/quantum-solver/. [Accessed: 17-Apr-2022].
[3] Tudorache, A. G., Manta, V. I., and Caraiman, S. (2021). Implementation of the Bernstein-Vazirani Quantum Algorithm Using the Qiskit Framework. Bulletin of the Polytechnic Institute of Iaşi. Electrical Engineering, Power Engineering, Electronics Section, 67(2), 31-40.
[4] Warke, A., Behera, B. K., and Panigrahi, P. K. (2020). Experimental realization of three quantum key distribution protocols. Quantum Information Processing, 19(11), 1-15.
[5] IBM, “Qiskit”. [Online]. Available: https://qiskit.org/. [Accessed: 17-Apr-2022].
[6] IBM, “IBM Quantum”, May-2016. [Online]. Available: https://quantum-computing.ibm.com/. [Accessed: 17-Apr-2022].
[7] IBM, “User account - IBM Quantum”, May-2016. [Online]. Available: https://quantum-computing.ibm.com/composer/docs/iqx/manage/account/#account-overview. [Accessed: 17-Apr-2022].
[8] IBM, “Learn Quantum Computation using Qiskit” [Online]. Available: https://qiskit.org/textbook/preface.html. [Accessed: 17-Apr-2022].
[9] IBM, “Single qubit Gates” [Online]. Available: https://qiskit.org/textbook/ch-states/single-qubit-gates.html. [Accessed: 17-Apr-2022].
[10] E. Bernstein and U. Vazirani, “Quantum Complexity Theory”, SIAM Journal on Computing, vol. 26, no. 5, pp. 1411-1473, 1997.
[11] IBM, “Bernstein-Vazirani Algorithm” [Online]. Available: https://qiskit.org/textbook/ch-algorithms/bernstein-vazirani.html. [Accessed: 17-Apr-2022].
[12] C. H. Bennett and G. Brassard, “Quantum cryptography: Public key distribution and coin tossing,” arXiv, 1984, doi: 10.48550/ARXIV.2003.06557. [Online]. Available: https://arxiv.org/abs/2003.06557 [Accessed: 17-Apr-2022].
[13] IBM, “Quantum Key Distribution” [Online]. Available: https://qiskit.org/textbook/ch-algorithms/quantum-key-distribution.html. [Accessed: 17-Apr-2022].
[14] W. Dür, S. Heusler, “What we can learn about quantum physics from a single qubit”, 06-Dec-2013. [Online]. Available: https://arxiv.org/pdf/1312.1463.pdf. [Accessed: 17-Apr-2022].