A publication of ISCA*:
International Society for Computers
and Their Applications

INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS

TABLE OF CONTENTS

* "International Journal of Computers and Their Applications is abstracted and indexed in INSPEC and Scopus."

Volume 25, No. 4, Dec. 2018 **ISSN 1076-5204**

International Journal of Computers and Their Applications

A publication of the International Society for Computers and Their Applications

ISCA Headquarters···•···P. O. Box 1124, Winona, MN 55987 USA···•···Phone: (507) 458-4517
E-mail: isca@ipass.net • URL: http://www.isca@isca-hq.org.

A Highly Secured Three-Phase Symmetric Cipher Technique

M. V. Praveen*
Southern Illinois University, Carbondale, IL USA

P. Majumder†
Indian Statistical Institute, Kolkata, West Bengal, INDIA

K. Sinha*
Southern Illinois University, Carbondale, IL USA

N. Rahimi*
Southeast Missouri State University, Cape Girardeau, MO USA*

B. Gupta*
Southern Illinois University, Carbondale, IL USA

Abstract

A novel three-phase symmetric cipher technique is presented in this paper. The main characteristic of the technique is that the length of cipher text can be made larger to any value of choice than that of plain text by adjusting the values of some key parameters. This difference in length complicates the statistical relationship between plain text and cipher text; thereby making the process of cryptanalysis extremely difficult. Effectively, it helps to fulfill the main design objective of enhancing to a good extent the degrees of confusion and diffusion, which are the main two properties of a secure cipher as identified by Claude Shannon. We have shown that complexity of the three-phase encryption algorithm from the viewpoint of cryptanalysis is very high. It can be made computationally infeasible by appropriate choice of the different values of the three-part key used in the design. Statistical testing using NIST also ensures the strength of the approach.

Key Words: Symmetric encryption, diffusion, confusion, three-phase encryption.

1 Introduction

Cryptography incorporates security in data transmission. Confidentiality, authentication, integrity, and nonrepudiation are the main security goals. Note that cryptanalysis is a form of attack on an encryption algorithm and it is based on properties of the encryption algorithm [6, 17]. Key-based cryptographic methods are classified as asymmetric (public key) cryptography and symmetric key cryptography. Asymmetric approach uses two keys: public and private keys. Public key is used for encryption by a sender and the private key, only known to a receiver, is used for decryption. Symmetric or conventional encryption is a form of cryptosystem in which encryption and decryption are performed using the same key [17]. Symmetric key cryptography has two classes: stream ciphers and block ciphers. A stream cipher encrypts plaintext one byte at a time; it may also be designed to operate on one bit at a time. In general stream ciphering techniques are fast and the hardware requirement is reasonably small [1, 19, 20].

A block cipher is an encryption/decryption technique in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Many symmetric block encryption algorithms, such as the Data Encryption Standard (DES), are based on a structure referred to as a Fiestel block cipher [4]. DES has been the most widely used block cipher technique until recently. At present, Advanced Encryption Standard (AES) is being used for commercial applications. Another prominent block cipher mode of operation is XTS-AES [8, 15]. It is an IEEE standard. It describes a method of encryption for data stored in sector-based devices where the threat model includes possible access to stored data by the adversary [17]. In this context, the area of cryptanalysis of block ciphers is worth mentioning. There are two basic approaches, viz., linear [9] and differential cryptanalysis [2, 11]. Linear cryptanalysis is a known plaintext attack that uses a linear relation between the inputs and the outputs of an encryption scheme to assign probabilities to possible keys in order to determine the most possible one. Non-linear cryptanalysis [7] is an improvement of linear cryptanalysis in that it decreases the number of texts needed to cryptanalyze the cipher. Differential cryptanalysis on the other hand analyzes the effect of particular differences in pairs of

* Department of Computer Science.
† ACM Unit.

plaintexts on the difference of the corresponding cipher texts. It utilizes the differences to assign probabilities to possible keys in order to determine the most possible one [18].

In this context, it may be mentioned that Shannon [16] introduced the terms diffusion and confusion to capture the two basic building blocks for any cryptographic system. Every block cipher is the result of a transformation of a block of plain text into a block of cipher text. The mechanism of confusion is to make the relationship between the statistics of the cipher text and the value of the encryption key as complex as possible to thwart attempts to discover the key, and the mechanism of diffusion is all about making the statistical relationship between the plain text and the cipher text as complex as possible in order to thwart attempts to discover the key. Fiestel cipher structure [4] employs these two mechanisms in the following way. This structure consists of a number of identical rounds of processing. Substitution and permutation are performed in each round; thereby achieving a high level of diffusion and confusion. It has been pointed out [14] that the two mechanisms have been so successful in designing highly secured block cipher techniques that these have become the cornerstone of modern block cipher design.

In another approach, a randomly generated one-time pad is used for encryption [3, 5, 10]. After every use, a one-time pad is discarded and in that way it achieves a very high level of security. Theoretically, it offers complete security. However, its implementation faces two fundamental difficulties; first, a huge key distribution problem exists since for every message to be sent, a key of equal length is needed by both sender and receiver and second, generating large quantities of random keys is really problematic.

The work presented in this paper is an extension of one of our recently published works [13]. Initial versions of the encryption and decryption algorithms can be found in [13]. In this work, the objective is to enhance the degree of confusion and diffusion to an extent to make the statistical relationship between the plain text and the cipher text as complex as possible in order to thwart attempts to discover the key. To fulfil the objective, we will consider a three-phase encryption technique in a way to make the cipher text lengthier than the plain text. Obviously, this difference in length complicates further the statistical relationship between plain text and cipher text; thereby making the process of cryptanalysis extremely difficult. It may be noted that as per our knowledge, most existing symmetric encryption techniques generate cipher text having the same length as that of a given plaintext. Also, the encryption process will be such that the output of phase 1 will be the input to phase 2, and the output of phase 2 will be the input to phase 3; thereby making the process of cryptanalysis even harder. Both encryption and decryption schemes will be shown to be fairly simple and efficient. They can be applied to both block-level and non-block-level encryption. It will be shown that complexity of the three-phase encryption algorithm from the viewpoint of cryptanalysis is very high. It can be made computationally infeasible by appropriate choice of the different values of the three-part key used in the design. We will show that statistical testing using NIST [12] also ensures the strength of the approach.

The paper is organized as follows. In Sections 2 and 3, we present the three-phase encryption scheme and the decryption scheme respectively. Section 4 discusses its complexity from the viewpoint of cryptanalysis and in Section 5 we have presented the results obtained after applying NIST to our approach. Section 6 draws the conclusion.

2 Encryption

The presented symmetric encryption technique consists of three phases. It uses a key that has three parts. During encryption, each part is used in its corresponding phase. Decryption also has three phases and the three parts of the key are used in reverse order. The three-part key is stated below with each part separated by a comma:

Key = { X, [x1, x2, x3, x4, …, xi, …, xn], Y }

- First part (used in phase 1), X – Any integer that satisfies, 2(p-1) <= X < 2p – 255 (p is any integer greater than 9)
- Second part (used in phase 2), [x1, x2, x3, x4, …, xi, …, xn] – A set of integers that satisfy, 2(p-1) - X <= xi < 2p - X - 255 (p is any integer greater than 9). ‘n’ can be any value greater than 1.
- Third part (used in phase 3), Y – some very large integer.

2.1 Phase 1

This Phase uses an integer, X as the key. To start with, each plain text character is converted to its corresponding ASCII value. The encryption process works as follows: the value of X is added to each of the ASCII values obtained from the plain text. Throughout the first two phases, any intermediate output value should not cross the interval [2(p-1), 2p), where we assume that for binary representation, all outputs have to be represented using the same p number of bits each, with p larger than 9. Choice of X is governed by the following condition:

$$2(p-1) <= X < 2p - 255 \ (p > 9), \text{ and } X >= 512 \qquad (1)$$

Justification for the above choice of the values of X is stated below.

Any value of X lies in the interval [2(p-1), 2p), where p is a positive integer. The value of X must not be in the range of (2p - 255, 2p). This is because if we select a value in that range the output of phase 1 will have a chance of getting a value greater than or equal to 2p; in such a situation, an output value will need more than p bits for its representation contrary to the assumption that each output value consists of p number of bits. The above argument validates the first part of condition (1), namely 2(p-1) <= X < 2p – 255.

The second part of condition (1) is that the value of X must be greater than or equal to 512. If we use any value less than 512, this second part does not satisfy (except for X=256). Let us see what happens if X is less than 512. Consider X = 365. For ASCII value 0, the output is 365 and for ASCII value 250, the output is 615. Note that X (= 365) needs 9 bits, i.e. p = 9;

however, the value 615 requires 10 bits, i.e. p = 10. This does not satisfy the assumption that all outputs have to be represented using the same p number of bits each. This problem can be avoided if X >= 512, which is the second part of condition (1).

Note that one of the main design characteristics of the present work is to enhance the amount of diffusion. Observe that the difference in length of cipher text from plain text is caused by the value of X because we are converting each 8-bit representation of an ASCII character to a value that needs at least p bits (p > 9). If we take a very large value for X, the cipher text length will even increase more. In other words, it enhances further the amount of diffusion; thereby making things even more complicated for the intruders.

An example: We use the following plain text and the three parts key to explain the working of the different phases. Here we show the working of the first phase. Let the plain text be 'Kumar' and the three-part key be:

Key: {1527, {212, -496, 45, -223}, 6543987}

The first part of the key is: X=1527. This value of X satisfies condition (1). In this phase, X is added to the ASCII value of each character of the plaintext. The first phase outputs are shown in the 3rd row of Table 1.

Table 1: Phase-1 output of the encryption scheme

Character	K	u	m	a	r
ASCII	75	117	109	97	114
X	1527	1527	1527	1527	1527
ASCII + X (OUTPUT)	1602	1644	1636	1624	1641

The pseudo code for phase 1 is stated below.

function **phase1** (plaintext, X);

Input: plaintext of length 'r' and first part of the key (X)
Output: An array of decimal integers [A_1, A_2, …, A_i, … A_r]
1. Create an array with the ASCII values of all the r plain text characters - [P_1, P_2, …, P_i, …P_r]
2. for i=1 to r
3. $A_i = P_i + X$;
4. return [A_1, A_2, …, A_i, … A_r]

2.2 Phase 2

Phase 2 uses second part of the key. The main idea of using this phase is to distribute the interval of outputs from phase 1, viz., [X, X+255] to a new and much larger interval [2(p-1), 2p). For this purpose, we use the second part of the three-part key as follows: the *i*th element of the vector [x1, x2, x3, x4, …, xi, …, xn] (n > 1) is added to the *i*th output value obtained from phase 1. If the number of output values from phase 1 is larger than the length of the vector, for the rest of the output values addition is repeated with the elements of the vector until all the output values have been added. This phase helps in creating more diffusion.

Now we state below how to calculate the interval for selecting the elements of the vector. This can be done using the logic that the output values of phase 1 belonging to the interval viz., [X, X + 255], when added with an element in the vector [x1, x2, x3, x4, …, xi, …, xn] should not cross the interval [2(p-1), 2p). So the valid interval for the elements of the vector is the difference of [X, X + 255] from the interval [2(p-1), 2p). Therefore, the resulting interval for selecting the value of any xi is [2(p-1) – X, 2p – X – 255). Let us check the validity of this interval. Consider the least possible value from the outputs of phase 1, i.e. X and the least value of the above interval, i.e. 2(p-1) – X. The addition of these two values results in the least value 2(p-1) of the interval [2(p-1), 2p). Similarly, the largest values from those two intervals when added together results in the value 2p of the interval [2(p-1), 2p). Hence it validates the choice of the interval [2(p-1) – X, 2p – X – 255).

In the example, X = 1527, so p = 11. From the values, the calculated range is [-503,266). Without any loss of generality, we have selected the number of elements of the second part of the key as 4; i.e. n is 4. The vector of four integers is [212, -496, 45, -223]. The above mentioned additions of the elements of the vector and the corresponding output values obtained from phase 1 has resulted in the phase 2 output as shown in Table 2.

Table 2: Phase-2 output of the encryption scheme

Phase 1 output	1602	1644	1636	1624	1641
Phase 2 key	212	-496	45	-223	212
Summation (OUTPUT)	1814	1148	1681	1401	1853

The pseudo code for phase 2 is stated below.

function **phase2** ([A_1, A_2, …, A_i, … A_r], {x_1, x_2, …, x_i, …, x_n});

Input: phase 1 output [A_1, A_2, …, A_i, … A_r] and second part of the key {x_1, x_2, …, x_i, …, x_n}
Output: An array of decimal integers [B_1, B_2, …, B_i, … B_r]

```
1.  Calculate n from the vector {x1, x2, …, xi, …, xn}
2.  Set z=0
3.  for i=1 to r
4.          if (n/i = 0)
5.                  z = n;
6.                  Bi = Ai + xz;
7.          else
8.                  z = i % n;
9.                  Bi = Ai + xz;
10. return [B1, B2, …, Bi, …Br]
```

2.3 Phase 3

In this phase, first, the integer outputs from phase 2 are converted to their respective binary formats. Then all these are concatenated to form a single binary string. Now consider blocks of size b such that 2(b-1) <= Y < 2b.

The first b bits of the concatenated binary string are XORed with the binary equivalent of Y, the third part of the three-part key. The result is used to XOR with the next b bits of the concatenated string. This process is continued till the end of string. The purpose of this phase is to make the changes to the bits to appear random to an attacker. It makes the process of cryptanalysis on the cipher text more difficult. Also, as mentioned earlier Y should be very large resulting in a large block size. Larger block size means greater diffusion; thereby it further enhances the level of security.

Let us continue with the example. The output of the second phase is a set of integers. The number of integers in the set is equal to the number of characters in the plain text. Now convert each integer into binary format of length p bits and concatenate them. They appear as follows.

1814 – 11100010110; 1148 – 10001111100; 1681 – 11010010001; 1401 – 10101111001; 1853 – 11100111101:

The concatenated string is: 111000101101000111 110011010010001101011110011110011101

The block size b is 23 corresponding to Y (= 6543987 = 101000111101101001110011).

To start with, the first 23 bits of the string is XORed with the binary value of Y. Let us denote the result as C1.

C1 = 11100010110100011111001 XOR 110001111011010011 10011 = 00100101011001010001010

Consider the next 23 bits of the string and XOR with the last output. The result is denoted as C2.

C2 = 10100100011010111100111 XOR 001001010110010 10001010 = 10000001000011101101101

Note that only 9 bits are remaining in the concatenated string. Use the first 9 bits of the last output to XOR with the remaining 9 bits of the concatenated string. The result is: C3 = 100111101 XOR 100000010 = 000111111. Now, concatenate C1, C2, and C3 to get the string 00100101011001010001010100000010000111011011010100011111 1

The string C1C2C3 is divided into pieces of 8 bits each and each piece is converted to its ASCII equivalent character. If length of the string is not divisible by 8, append the required number of 0's at the end. In the example the length of the binary string is 55. So, one 0 is added at the end to make the length 56, a multiple of 8. The final output string of third phase is: 00100101011001010001010100000010000111011011010001111110

Observe that the final output (cipher text) in our example has 7 characters which is more than the size of plain text which is 5 characters long. Thus, we make sure that cipher text is always lengthier than the corresponding plain text. The pseudo code for phase 3 is stated below.

2.4 Cipher Text Length

Now we discuss how the length of cipher text varies from the length of plain text and also which part of the three-part key is responsible for this variation. As can be seen from the encryption scheme that the first phase key takes the ASCII values and converts them to large numbers by adding X to each ASCII value of the plain text characters. The second phase key just distributes the values from [X, X+255] to [2(p-1), 2p]. So, the second phase key doesn't contribute to the change in the length of cipher text.

function **phase3** ([B_1, B_2, …, B_i, … B_r], Y);

Input: phase 2 output [B_1, B_2, …, B_i, … B_r] and third part of the key Y
Output: Cipher text

1. Convert each decimal value [B_1, B_2, …, B_i, … B_r] to least possible number of binary bits
2. Concatenate all those binary bits to form long binary string L
3. Calculate b from $2^{(b-1)} <= Y < 2^b$
4. D is initialized with binary value of Y
5. for each b bit block, C_i in the long binary string L
6. $O_i = C_i$ & D;
7. D = O_i;
8. Concatenate all b bit blocks of O_i
9. Append 0s to make the length of binary string in the previous step divisible by 8
10. Convert each 8 bits of the string in previous step to a character using ASCII table.
11. The result is cipher text.

The third phase key Y is only used for XOR operation and so it does not affect the length of cipher text. Therefore, it is seen that only the value of X contributes to the change of length of cipher text.

For quantitative evaluation of the changes in cipher text length caused by the choice of the values of X, we have used different values of X and calculated the 'length ratio' of cipher text and plain text for each X. The following relation holds among X, length of plain text (LP), and length of cipher text (LC).

$LC = (p/8) * LP$, where $2(p-1) <= X < 2p$

Note that length of cipher text is 'p/8' times the length of plain text. As X is always greater than 512, length of cipher text is always greater than length of plain text. The graph in Figure 1 shows the effect of X on LC/LP. Note that the curve is a logarithmic one. This is because the value of LC/LP depends linearly on the number of bits, p and p is calculated from the log value of X.

Note that the increase in cipher text length will result in the increase in size of the encrypted file. This is true however, we are able to make it more secure and it will make up for the increase in the size.

Formal presentations of the proposed encryption and decryption algorithms appear in Figures 2 and 3.

3 Decryption

Decryption is done in the reverse order of encryption that is, we first perform the inverse operations of the third phase of encryption, then second, and then first. Y, the third part of the three-part key is used in the first phase of decryption and the first part X in used in third phase. We use the same example considered for encryption to explain the working of the different phases of decryption.

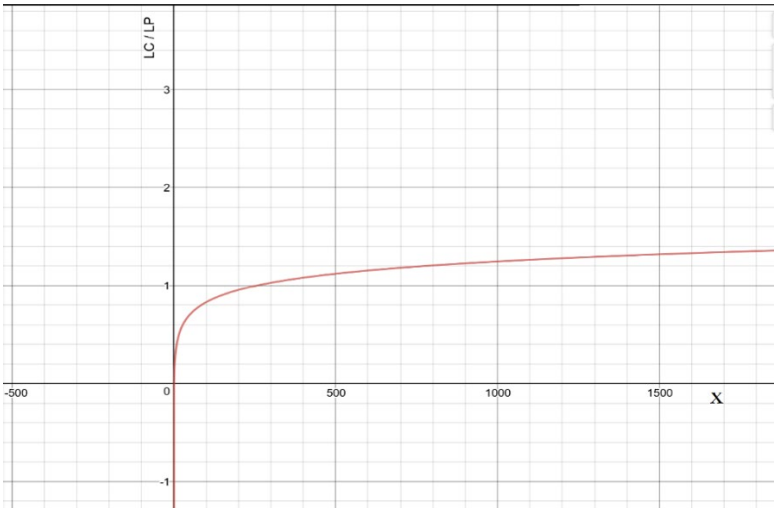


Figure 1: Effect of X on LC/LP

X	First Phase key, $2^{(p-1)} <= X < 2^p - 255$ $(p > 9)$
$\{x_1, x_2, x_3 \ldots, x_i \ldots, x_n\}$	Second Phase key, $2^{(p-1)} - X <= x_i < 2^p - X - 255$ $(p > 9)$
Y	Third Phase key, should be a very large value
p	$2^{(p-1)} <= X < 2^p$
b	$2^{(b-1)} <= Y < 2^b$
$P_1, P_2, P_3, \ldots P_i \ldots P_r$	P_i – ASCII value of ith plain text character
n	Number of elements in second phase key $(n > 1)$
$A_1, A_2, A_3, \ldots A_i \ldots A_r$	Phase-1 output
$B_1, B_2, B_3, \ldots B_i \ldots B_r$	Phase-2 output
L	Long Binary String
$C_1, C_2, C_3, \ldots C_i \ldots$	b bit parts of long binary string
$C_1', C_2', C_3', \ldots, C_i' \ldots$	b bit output after XOR

Phase-1

$A_i = P_i + X$ Input: P_i, Output: A_i $(1 <= i <= r)$

Phase-2

$B_i = A_i + x_j$, $j = \begin{cases} n & \text{if } i \mid n \\ i \bmod n & \text{otherwise} \end{cases}$ Input: A_i, Output: B_i $(1 <= i <= r)$

Phase-3

Convert decimal values of all Bi values to binary and then concatenate them to form a long binary string L. The binary string is divided into b bits each and perform XOR with the value of Y. The process is continued using Cipher Block Chaining (CBC) until all the bits in long binary string are completed. Only remaining bits are XORed in the last step

$C_1 \oplus Y = C_1'$

$C_2 \oplus C_1' = C_2' \ldots.$

$C_i \oplus C_{i-1}' = C_i'$ Input: B_i, Output: Cipher text

Concatenate all C_i' values, divide into 8 bits each. Convert each 8 binary bits to corresponding ASCII value. The result is the cipher text.

Figure 2a: The encryption algorithm

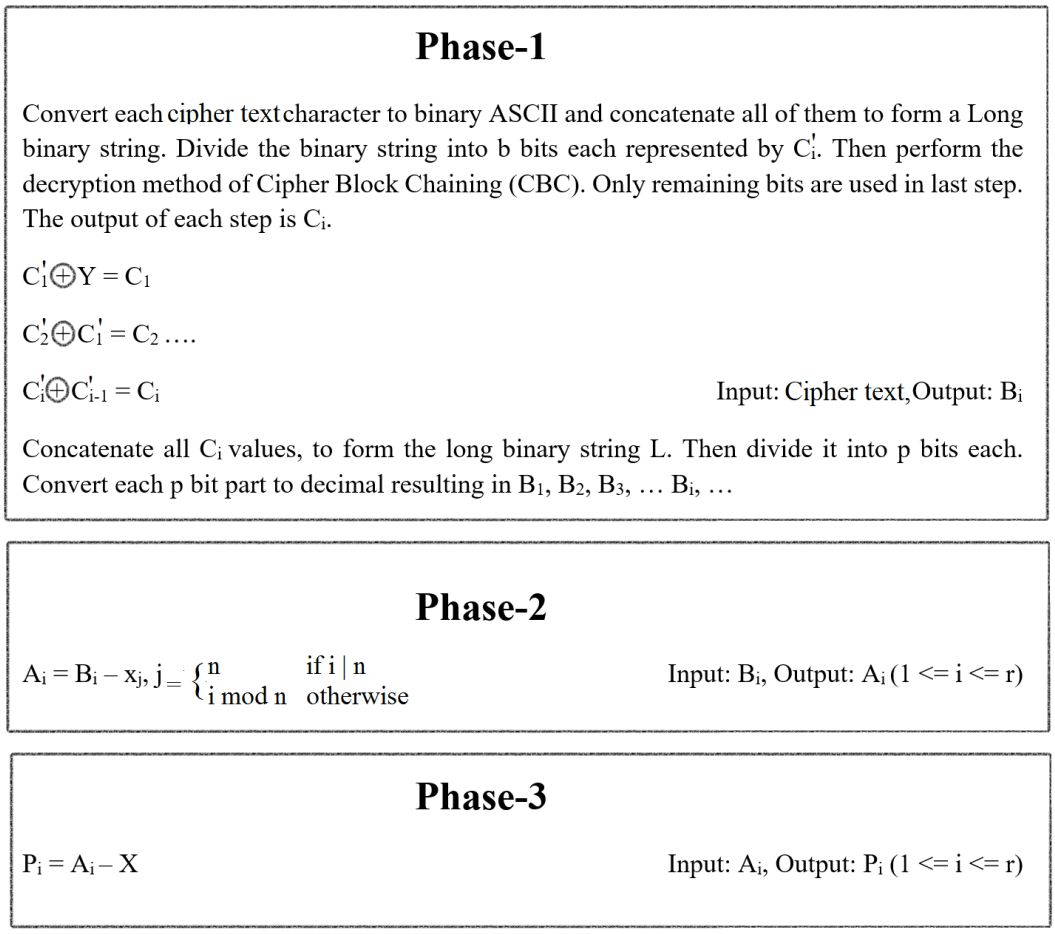


Figure 2b: The decryption algorithm

3.1 Phase 1

Step 1: Convert each character of cipher text to 8 bit binary and then concatenate them to form the long binary string.

Step 2: Calculate the remainder of the concatenated string by dividing the length of the concatenated binary string by p.

Step 3: Remove the remaining number of bits from the end of the string.

Step 4: perform the XOR operation using the binary value of Y with b bits as block size.

Step 5: Convert the cipher text into a single binary string.

Step 1 generates the following binary string, which is the same as the output of phase 3 in encryption. The binary string is: 00100101011001010001010100000010000111011011010001111110

The length of this string is 56. The value of p is 11. The remainder for 56/11 = 1. Now, remove those extra bits, which are appended in the final phase of encryption; it is the last 0 bit here. So the execution of steps 2 and 3 results in the binary string: 00100101011001010001010100000010000111011011001100111

Next, we execute step 4 as follows. Perform XOR operation between the binary value of Y and the above string. The equivalent binary string for Y is 11000111101101001110011 and its length is 23. So, with block size 23, perform XOR operation with the binary string obtained from step 3. As in encryption, first consider the first 23 bits of the binary string and perform XOR with the binary value of Y. Let the resulting string after the XOR operation be denoted as C11. It is shown below.

C11 = 00100101011001010001010 XOR 11000111101101001110011 = 11100010110100011111001

Now, consider the next 23 bits of the binary string obtained from step 3 and XOR with the previous 23 bits of the same binary string. The result is shown below.

C12 = 10000001000011101101101 XOR 00100101011001010001010 = 10100100011010111100111

Only 9 bits are remaining in the binary string obtained from Step 3. The remaining 9 bits of binary string are XORed with the first 9 bits of the previous 23 bits of long binary string.

C13 = 000111111 XOR 100000010 = 100111101

Now, Step 5 is executed; that is, concatenate C11, C12, and C13. The resulting string is the output of phase 1 of decryption and input to phase 2 of decryption and is written below:

11100010110100011111001101001000110101111001111001
11101

3.2 Phase 2

Step 1: Divide the binary string obtained from phase 1 into blocks of p bits each.

Step 2: Convert each p bit-block into its corresponding decimal value.

Step 3: Then the corresponding xi element of the second part of the key is subtracted from each of these decimal values.

Thus, the result of steps 1 and 2 is as follows:

11100010110 = 1814; 10001111100 = 1148; 11010010001 = 1681; 10101111001 = 1401; 11100111101 = 1853;

Step 3 generates the values shown in the third row of Table 3. These are the inputs to phase 3 of decryption. Note that we perform similar steps as in phase 2 of encryption, but we use the difference here.

3.3 Phase 3

Phase 3 uses X. The value of X is subtracted from each of the decimal values obtained from the output of phase 2 to get the ASCII values of the characters of the plain text. Finally converting those ASCII values to text will result in the plain text. It is shown in Table 4. The process of decryption is completed and the resultant plain text is 'Kumar'.

4 Complexity

We compute the complexity of our three-phase encryption algorithm from the viewpoint of cryptanalysis to show how difficult it would be to break in.

From the viewpoint of cryptanalysis, complexity involved in discovering the value of X is O(2p), because $2(p-1) <= X < 2p - 255$. Complexity involved in discovering the values of the n elements of the second part of the three-part key is O(2n*(p-1)), because the length of the interval for selecting the n elements of the second part of the key is [2(p-1) – X, 2p – X – 255) and the values of the n elements depend on the choice of X. Next, complexity of discovering the Y value is O(2b), since Y lies in the interval [2(b-1), 2b). Therefore, Complexity of the three-phase encryption algorithm from the viewpoint of cryptanalysis = O(2p) * O(2n*(p-1)) * O(2b) = O(2((n+1) * p + b - n)).

In the example used in this work, this complexity is O(2((4+1) * 11 + 23 - 4)) = O(274). The noteworthy point of the proposed algorithm is that the value of n is user's choice and has a great impact on complexity. We can attain very high complexity easily by selecting very large values for X, Y, and n; thereby making the process of cryptanalysis computationally infeasible.

5 Statistical Testing

There exist few statistical test suites in the literature [12]. These are mainly the DIEHARD suite, the Crypt-XS suite, and the NIST statistical test suite. Each of these defines different sets of parameters for statistical testing the different randomness properties of a given binary sequence. Among these, the NIST statistical test suite is the most widely used one

Table 3: Phase-2 output of Decryption

Phase 1 output in decimal values of p bits each	1814	1148	1681	1401	1853
Phase 2 key	212	-496	45	-223	212
Difference (OUTPUT)	1602	1644	1636	1624	1641

Table 4: Phase-3 output of Decryption

Phase 2 output	1602		1644	1636	1624	1641
X (phase 3 key)	1527		1527	1527	1527	1527
Step 2 – X (ASCII)	75		117	109	97	114
Corresponding character	K		u	m	a	r

in the field of cryptography. Basically, for a given binary sequence s it needs to establish whether or not s passes or fails a statistical test. The evaluation approach of NIST is all about computing a test statistic for the sequence s and its corresponding probability value (P-value) which then is compared with a fixed significance value α. Note that P-value ∈ [0, 1] and α ∈ (0.001, 0.01]. Success is declared whenever P-value ≥ α; otherwise failure is declared. NIST uses this approach due its flexibility. For a clearer understanding, we now state characteristics of some of the NIST statistical tests. For example, Frequency test detects defect if there are too many zeroes or ones; Cumulative Sums detects defect when there are too many zeroes or ones at the beginning of the sequence; Linear Complexity detects defect if there is a deviation from the distribution of the linear complexity for finite length (sub)strings. The details of all the tests can be found in [12].

As a means to observe the strength of the proposed approach we have used NIST statistical Test suite [12] to obtain the results of different statistical tests. The results for the various statistical tests are shown in Table 5. Note that all these results are very close to 1. It implies that our approach has passed all the tests from the viewpoint of randomness.

Table 5: Randomness test results

Test	Cipher Text
Frequency	0.97
Block Frequency	0.98
Cumulative Sums	0.98
Runs	0.96
Longest Run	0.99
Rank	0.96
FFT	0.98
Nonoverlapping Template	0.97
Overlapping Template	0.99
Universal	0.99
Approximate Entropy	0.96
Random Excursions	0.97
Random Excursions Variant	0.98
Serial	0.97
Linear Complexity	0.98

6 Conclusion

A novel three-phase symmetric cipher technique has been presented. Proposed encryption and decryption schemes are fairly simple and efficient, and can be applied to both block-level and non-block-level encryption. The main feature of the technique is that the length of cipher text can be made larger to any value of choice than that of plain text by adjusting the values of some key parameters. Obviously, this difference in length complicates further the statistical relationship between plain text and cipher text; thereby making the process of cryptanalysis extremely difficult. We have shown that complexity of the three-phase encryption algorithm from the viewpoint of cryptanalysis is very high. It can be made computationally infeasible by appropriate choice of the different values of the three-part key. Besides, the strength of the proposed approach has been verified with the help of NIST statistical Test suite. It can be very well argued that the extra bits needed to generate a cipher text will not at all affect efficient use of memory and bandwidth; rather the computationally infeasible feature is what is needed in this era of highly secured communication.

References

[1] M. Abumuala, O. Khalifa, and A. H. A. Hashim, "A New Method for Generating Cryptographically Strong Sequences of Pseudo Random Bits for Stream Cipher," *Proc. IEEE Int. Conf. Computer and Communication Engineering*, pp. 1-4, May 2010.

[2] E. Biham and A. Shamir, "Differential Cryptanalysis of the Full 16-Round DES," *Proc. Crypto'92*, LNCS, Springer-Verlag, 740: 487-496, 1992.

[3] M. Borowski and M. Lesniewicz, "Modern Usage of 'Old' One-Time Pad," *Proc. IEEE Int. Conf. Communication and Information Systems*, pp. 1-5, Oct. 2012.

[4] H. Fiestel, "Cryptography and Computer Piracy," Scientific American, May 1973.

[5] C. H. Huang and S.C. Huang, "RFID Systems Integrated OTP Security Authentication Design," *Proc. IEEE Int. Conf. Signal and Information Processing Association Annual Summit and Conf.*, pp. 1-8, Nov. 2013.

[6] Gary C. Kessler, "An Overview of Cryptography," 2016.

[7] L. R. Knudsen and M. J. B. Robshaw, "Non-Linear Approximations in Linear Cryptanalysis," LNCS, Springer-Verlag, 1070: 224-236, 1996.

[8] M. Liskov, R. Rivest, and D. Wagner, "Tweakable Block Ciphers," Advances in Cryptography-Crypto'02, LNCS, Springer-Verlag, 2442:31-46, 2002.

[9] M. Matsui, "The First Experimental Cryptanalysis of the Data Encryption Standard," Advances in Cryptology: *Proc. Crypto'94*, pp. 1-11, 1994.

[10] C. Matt and U. Maurer, "The One-Time Pad Revisited," *Proc. IEEE Int. Conf. Information Theory*, pp. 2706-2710, July 2013.

[11] S. Murphy, "The Cryptanalysis of FEAL-4 with 20 Chosen Plaintexts," *Journal of Cryptology*, 2(3):145-154, 1990.

[12] National Institute of Standards and Technology, An Introduction to Computer Security: The NIST Handbook, Special Publication 800-12, Oct. 1995.

[13] M. V. Praveen, B. Gupta, and K. Sinha, "A Novel Three Phase Symmetric Cipher Technique," *Proc. CATA 2017*, Honolulu, pp. 137-142, April 2017.

[14] M. Robshaw, "Block Ciphers," RSA Laboratories Technical Report TR 601, August 1995.

[15] P. Rogaway, "Efficient Instantiations of Tweakable Block Ciphers and Refinements to Modes OCB and PMAC," Advances in Cryptology-Asiacrypt 2004, LNCS, Springer-Verlag, 329:16-31, 2004.

[16] C. Shannon, "Communication Theory of Secrecy Systems," *Bell Systems Technical Journal*, No. 4, 1949.

[17] William Stallings, "Cryptography and Network Security Principles and Practice," Fifth Edition, 2011.
[18] F. X. Standaert, G. Piret, and J. J. Quisquater, "Cryptanalysis of Block Ciphers: A Survey," UCL CryptoGroup, 2003.
[19] K. M Sujatha, N. Sivakanya, K. Srikanth, R. Shetty, and P.V. A. Mohan, "A Review of some Recent Ciphers," *Proc. IEEE Int. Conf. Circuits, Controls and Communications*, pp. 1-6, Dec. 2013.
[20] T. D. B. Weerasinghe, "An Effective RC4 Stream Cipher," *Proc. IEEE 8th Int. Conf. Industrial and Information Systems*, pp. 69-74, Dec. 2013.

M. V. Praveen received his MS degree in Computer Science from Southern Illinois University, Carbondale in 2016. He is now pursuing his Ph.D. degree in Computer Science at Missouri University of Science & Technology. His current research interest is designing threat models for Advanced Metering Infrastructure (AMI) and Vehicular Networks.

Pratham Majumder is pursuing his Ph.D. degree in Computer Science and Technology from University of Calcutta, India. He obtained his M.Tech degree with specialization in Communication Engineering from VIT University, India in 2014 and B.Tech degree in Electronics and Communication from West Bengal University of Technology, India in 2012. Since November 2013, he has been with the Indian Statistical Institute, Kolkata as a Project Personnel in the Advanced Computing and Microelectronics Unit. His research interests include energy-efficient wireless communication and security. Currently, he is a faculty member of Computer Engineering in Indrashil Institute of Science and Technology, India.

Koushik Sinha is currently an Assistant Professor in the Department of Computer Science at Southern Illinois University, Carbondale. He leads the Complex Networks and Applications research group at SIU, Carbondale. He is the co-author of the book Wireless Networks and Mobile Computing published by CRC Press, Taylor and Francis Group, USA in 2015. Prior to joining SIU, he was with the *Social Computing Group* of *Qatar Computing Research Institute*, Qatar from 2013 to 2015. Previously, he was a Research Scientist at *Hewlett-Packard Labs*. He also spent 7 years with *Honeywell* as a Lead Research Scientist. He received the Young Scientist Award from the Indian Science Congress Association (ISCA) in 2009 and the N. V. Gadadhar Memorial Award from the Institution of Electronics and Telecommunication Engineers (IETE) in 2011. He was the IEEE ANTS (the only conference in India which is financially sponsored by IEEE Communications Society) TPC Co-chair in 2016 and 2017. He is also the General Co-chair for IEEE ANTS 2018. He is a senior member of the IEEE. His current research focus is in the areas of 5G, wireless sensor networks, IoT and Fog Computing.

Nick Rahimi is an Assistant Professor at Computer Science Department at Southeast Missouri State University. His main research interests revolve around Computer and Network Security, Distributed Systems, Peer-to-Peer Networks and their Privacy, and Data Communication. He has earned two Bachelor of Science degrees in Software Engineering and Information Systems Technologies. Nick obtained his Master and Ph.D. degrees in Computer Science from Southern Illinois University.

B. Gupta is a Professor at the Department of Computer Science, Southern Illinois University at Carbondale. He is a senior member of IEEE. His current research interest includes fault tolerant mobile computing, P2P network architecture, communication protocols, and security.

AVISTED—Analysis and Visualization Toolset for Environmental Data

Likhitha Ravi*, Eric Fritzinger*, Sergiu M. Dascalu*, Frederick C. Harris, Jr*.
University of Nevada, Reno, NV, USA

Abstract

Data analysis and visualization are two effective ways to unveil possible rules hidden in the environmental data. Environmental data is usually very large and can be stored in a format not commonly used, such as NetCDF. Most existing data visualization and interaction tools cannot handle environmental data efficiently because of these characteristics. Furthermore, most existing environmental tools require a user to download the data and execute software locally. This is not very convenient if the user's computer does not have high computing power and the data size is very big. To address these concerns, in this paper we present AVISTED, a generic web-based toolset used by the environmental scientists to analyze, convert, extract and visualize large datasets. This approach involves executing most of the operations on the server side, provides different visualization techniques and supports multiple environmental data formats such as NetCDF and HDF5. Additionally, in this paper we also present a new approach for software design, GUI-Enhanced Activity Diagrams (GEAD) that links UML activity diagrams to user interface snapshots. As such, it enhances the collaboration between teams and reduces the development time. This paper provides detailed descriptions of the design aspects of AVISTED by leveraging the concepts and strategies of GEAD.

Key Words: GUI-enhanced UML activity diagrams; large datasets; software design; data visualization; web application.

1 Introduction

Nowadays, data is produced at unprecedented rates. In particular, in environmental sciences, climate data is being collected, stored and transformed using various modeling techniques to predict the future climate. For example, The Nevada Research Data Center (NRDC) [18] stores sensor-based climate data of Nevada and the modeled datasets. The National Oceanic and Atmospheric Administration (NOAA) [17] provides the data about the weather and climate in the US. Similarly, Cal-Adapt [9] provides the climate information of California. Although the climate information is easily available, it is hard to get an insight into the data without specialized analysis and visualization tools.

The most common steps performed by a climate scientist are extracting data, applying models on the extracted data, converting the resulting model to the data format which is supported by the visualization tool, and visualizing it using data analysis tools, such as Matlab and R [28]. However, most of these require a user to install the software on the computer locally and some tools even require programming skills. These can be a problem for scientists who would like to analyze a large amount of data with a standard personal computer and do not know how to program. In order to address these concerns, in this paper we present a new approach entitled Analysis and Visualization Toolset for Environmental Data (AVISTED) aimed at addressing scientific data analysis needs.

AVISTED is a web-based visualization toolset. It helps climate researchers to find the trends, changes of variables with time, and dependencies between variables. AVISTED incorporates features such as User Authentication, User Authorization, Model Output Management, Model Operation, Tools Upload, and Archives. It supports datasets in NetCDF, HDF5, ASCII and CSV formats. Users can perform extraction, visualization, view, download and save operations on a dataset. Visualizations that are developed using the toolset can also be saved in the user accounts for accessing them in the future. Also, to support extensibility users are allowed to upload their own tools for data processing with the administrator's approval. AVISTED is implemented using the client-server architecture. It executes all the operations except visualization on the server side, which reduces the computing burden on the client side. Therefore, even if a normal computer is used, complex operations can be performed efficiently.

When we designed a supporting toolset for the AVISTED approach, a new design method titled GUI – Enhanced Activity Diagrams (GEAD) has been used. More details about GEAD is presented in our previous work [23]. The purpose of GEAD is to provide a comprehensive view of the application to the developers and testers. GEAD provides a user interface design for each significant activity in a UML activity diagram. This approach helps the developers in identifying activities that are unique and common between the user interfaces. Identifying these connections at design time are very helpful in the applications that are implemented using the MVC architecture. Furthermore, this technique helps users and stakeholders in getting a meaningful link between the execution and the user interface. GEAD is applicable in many fields such as web development and video game development. In this paper, we

* Department of Computer Science and Engineering. College of Engineering. Email: {ravi, ericf, dascalu, fred.harris}@cse.unr.edu.

showcase the GEAD approach by using its concepts in designing the activity diagrams of AVISTED.

The rest of the paper is structured into six sections. Section 2 presents the background from the scientific perspective. Section 3 gives an overview of AVISTED's software requirements. Section 4 presents the new GEAD technique and its usage in designing AVISTED. Section 5 presents an application scenario of AVISTED. Section 6 provides a comparison of AVISTED with related tools. Lastly, Section 7 concludes the paper by outlining several directions of future work for the AVISTED methodology.

2 Background on Scientific Need

AVISTED has been created to address the scientific needs of the environmental researchers. It is now integrated with NRDC 0 and is in the testing phase. The application uses a web interface to perform its functionalities by interacting with RESTful APIs for data conversion and data extraction. After moving the AVISTED toolset to the NRDC server, we have conducted functionality testing, use-case based testing, and interface testing to check if the application is functioning as per the specifications, navigating across different pages correctly, and connecting through the application layers, respectively.

Furthermore, in the future we intend to conduct more testing of the application with the participation of the end users in order to obtain valuable user feedback. Also, it is essential to evaluate the performance of the application with an increased number of concurrent users. However, as the application uses the RESTful API's for time intense tasks such as data extraction and conversion, these components can be placed in different Docker containers as needed. And, to improve the performance when the number of users increases, the computation tasks can be distributed to worker containers and processed in parallel using a job queue as discussed in our previous work [29] which mainly presents a new elastic scale hybrid server method for dynamically adjusting the web hosting resources based on queueing theory and user feedback.

This section gives the background of NRDC and explains why we developed the AVISTED toolset.

2.1 The Nexus Project & NRDC

NRDC serves in a critical role of science cyberinfrastructure for sensor-based and modelled data management. It facilitates the acquisition, transport, storage, query, and dissemination of observational data created by automated digital sensor systems. The NRDC participates in cutting-edge software and system development to enhance the next-generation science that leverages the Internet of Things (IoT). The goal of NRDC is to transform the scale, quality, impact, and bottom-line cost of research projects in Nevada that seek to deploy automated sensor systems as part of their scientific workflow. NRDC is dedicated to providing services and curation for project-level datasets and needs 0. The center was born out of a data portal that was developed during a Track 1 NSF EPSCoR project on climate change that included a cyberinfrastructure component called Nevada Climate Change Portal (NCCP). NCCP has been developed with support from a large, 5-year (2008-2013) NSF EPSCoR Research Infrastructure Improvement (RII) grant to provide data resources and computing support for scientists researching the long-term effects of climate change in Nevada. To achieve its goals, NCCP contains not only environmental datasets but also software tools and services created by Nevada computer engineers to help data acquisition and processing. One such tool is the new web application AVISTED that we developed, which enables data selection, extraction, download, conversion, and visualization of the modelled environmental datasets.

NRDC is currently funded exclusively through NSF-EPSCoR programs as part of large interdisciplinary research efforts 0. NRDC operates remote high-speed digital data links as part of the Nevada Seismological Laboratory wide-area research data network. Part of the cyberinfrastructure team of the NEXUS project [6], we maintain a cluster of servers that house its primary data storage and software services. NRDC maintains near-real-time archival of both flat-file and relational data structures of small-to-moderate scale. Its individual project databases range from a few hundred to billions of individual data points. NRDC exposes project data to researchers and collaborators using high-performance query interfaces as well as aggregated file archives. The data is shared directly with the Western Regional Climate Center (WRCC), the Southern Nevada Research Center at UNLV, the PRISM Climate Group at the Oregon State University, and the DataONE distributed data curation clearinghouse [8]. The NRDC develops solutions for research cyberinfrastructure (CI). The CI team members are active participants in the Earth Science Information Partners (ESIP) Federation, and work as national community members to improve standards, practices, and technologies in scientific data management. Their research includes a significant student education and training component, where computer science and engineering student's interface with domain scientist "clients" and seek to aid their sensor-based research using CI tools. We believe that this cross-disciplinary integration of engineering with domain science can ultimately lead to higher-quality science and research in Nevada.

2.2 Need of Scientists

NRDC provides scientists with quality data but the scientists also need tools to further explore the data and find valuable rules and patterns within the data. Extraction, visualization, and conversion are three common data analysis operations in the scientific field. Data extraction allows a user to extract a certain parameter from the dataset based on a particular time slot or location. This operation avoids loading irrelevant data and simplifies the data analysis. Data visualization can highlight the data trend of a parameter and display possible relations between two parameters. It is very important to find "gold nuggets" from the raw data 0. Data conversion is another significant operation. It means converting a data from one format into another format. A scientist usually uses multiple tools to analyze a dataset and these tools may each require input in a different file format.

Therefore, the scientist needs to write glue codes or use a data conversion tool. Since most scientists that use NRDC are environmental scientists and do not necessarily know how to program, data conversion is implemented in AVISTED for their convenience. Usually, the environmental scientists need to process a large amount of data and this requires a powerful machine or cluster. To reduce the burden, a client-server architecture is leveraged. The architecture does not require a powerful machine on the client side because most of the operations are executed on the server side. Our previous work proves that this architecture works well in this scenario [14], [20], and [27].

One use case of AVISTED is that a scientist can find possible connections between two environmental variables by extracting and drawing scatter plots. For example, the scientist can extract humidity and temperature values of the last ten years of Snake Range West Montane from a huge NetCDF file and visualize these two variables in a scatter plot by using AVISTED. Because the extraction step is done on the powerful server side, the operation is executed very fast even if the file size is huge. If the humidity and temperature variables are highly correlated with each other, the scatter plot should be approximately in a linear relationship, which means if one of these two parameters are known, a scientist can estimate the other parameter by using the correlation relationship.

The users of AVISTED are researchers, educators, students, and the general public. Using AVISTED, researchers will be able to quickly find the trends and patterns in data. Educators can use the toolset in the classroom while teaching the students about climate change. It also helps the general public in getting the information about the impacts of climate change.

3 Software Requirements

Initially, the main goal of the project was to visualize the climate modeling data provided by the Nevada Climate Change Portal. In this process, we first developed the Visualization Toolset for Environmental Data (VISTED). VISTED [24] is a non-generic version of AVISTED with limited functionalities. This tool operates on the output of a hind cast NCAR/WRF-based model developed by Dr. John Mejia at the Desert Research Institute, Nevada [16]. This output contains daily values for six environmental variables and extends over 30 years (1980–2009), so it is fairly large, and data operations like the ones mentioned above are needed by scientists who investigate the results of running this model. Extension plans of VISTED have led to the development of AVISTED that include expanding it to operate on a broader range of model outputs, adapting it to individual user preferences, and incorporating more diverse visualization capabilities.

Figure 1 shows the traditional activity diagram of VISTED with the actions involved in displaying the data extracted by the VISTED based on the user's selection. User selects the required variables/parameters, time period, location and submits the form. The user request is then sent to the application server. If the user input is valid, data is extracted and presented to the user or else an error message will be sent to the user. User can either

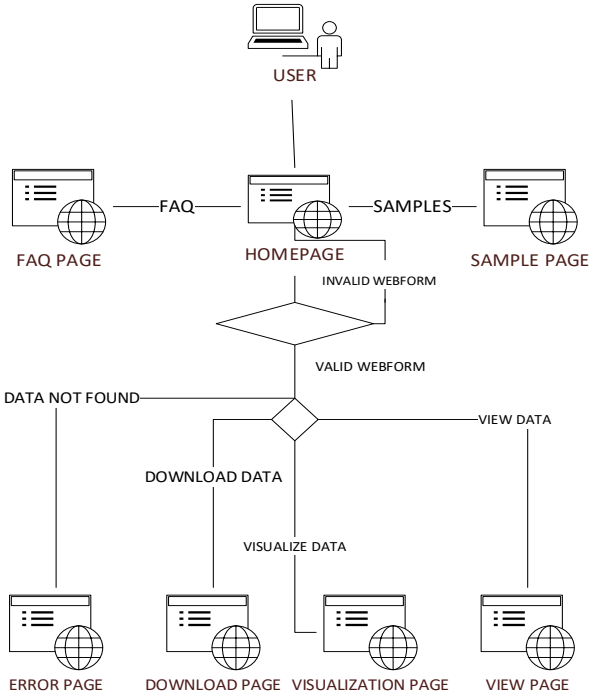


Figure 1: Traditional activity diagram of VISTED

view, download or visualize the extracted data.

Although VISTED helped environmental scientists in analyzing the NCAR dataset, based on the interviews conducted with researchers, an application that supports different file formats, modeled datasets and visualization techniques is much more beneficial for the broader community. Currently, a single generic tool that provides all of these features is not available for environmental researchers. We aim for the AVISTED approach to address these current challenges. AVISTED is an extended version of VISTED. The extension facilitates the application to be applicable to other model datasets in environmental sciences.

The new approach, AVISTED, for data analysis and visualization of environmental data incorporates: (i) a combination of key features that include extraction of datasets based on user requests, conversion of the extracted datasets to various data formats, a flexible user-friendly interface for visualizing the extracted data with several visualization options including zoom in and filter, capability to upload and manage model datasets represented in diverse data formats, and facilities to archive and download the work of authenticated users; (ii) environmental datasets; (iii) support for variety of environmental data formats such as NetCDF, HDF5, CSV, and ASCII; (iv) extensibility of the supporting environment, both in terms of the flexibility of the steps users can take for analyzing the datasets and in terms of available data processing features; and (v) applicability to other domains where geospatial information is represented using time series charts.

AVISTED has four types of users. A new user, registered user, admin user, and guest user. A new user will be able to register and log in to the application. A registered user can

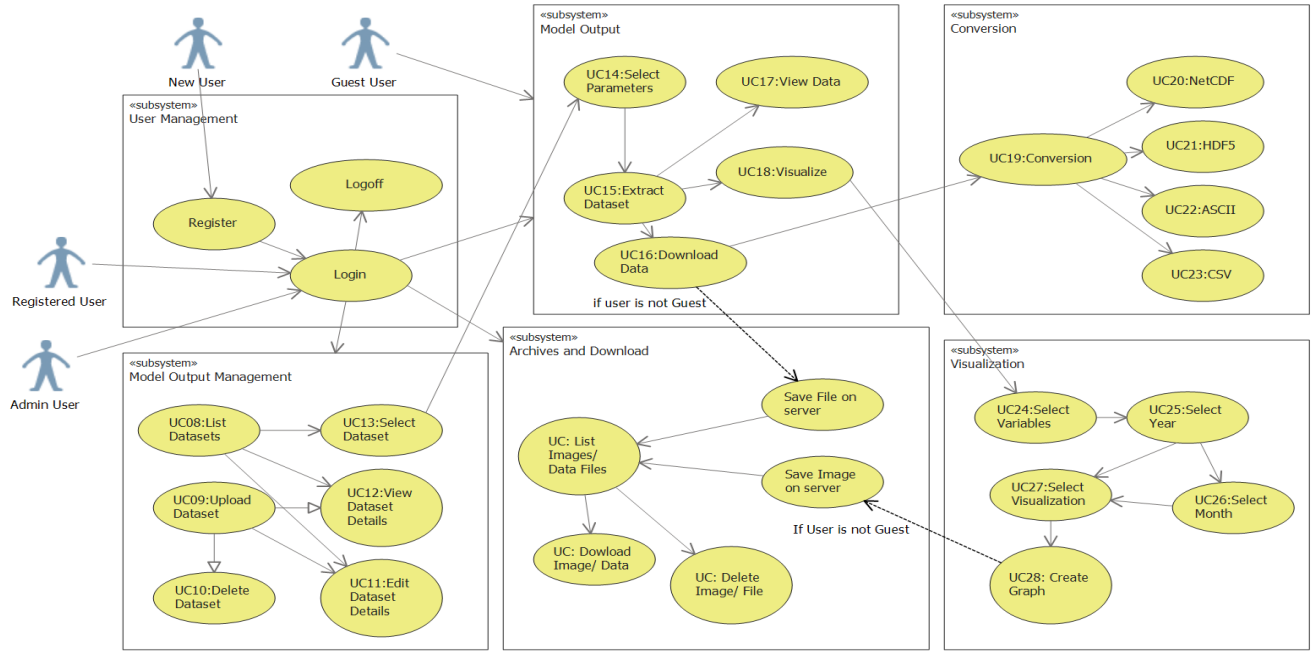


Figure 2: Use case diagram of AVISTED

access most of the features except some functionalities such as deleting a dataset uploaded by another user. On the other hand, a registered user can delete a dataset if it is uploaded by him/her. An admin has access to the all the features of the application. Finally, guest users have very limited accessibility to the application. They can only access the Model Output page. Figure 2 shows the use case diagram which depicts the interaction of different users with AVISTED. Each use case defines an action that AVISTED performs in collaboration with users.

The key requirements of AVISTED are noted based on the interviews conducted with individuals from different disciplines. Each requirement is given a unique ID and description. Non-functional requirements specify the standards that are used in evaluating the functionalities of the system. The main non-functional requirements of AVISTED are listed in Table 1.

Functional requirements define the functions of the software system. The main functional requirements of AVISTED are presented in Table 2.

Table 1: Non-functional requirements of AVISTED

ID	Description
T01	AVISTED shall be platform independent.
T02	AVISTED shall support different browsers.
T03	AVISTED shall support devices like desktops, laptops, tablets, and mobile phones.
T04	AVISTED shall be scalable.
T05	AVISTED shall be extensible and reusable.

4 Architecture and Design

4.1 Design Solution

AVISTED is a web application that uses a 3-tier client-server architecture. The first tier holds the presentation logic, the second tier the business logic, and the third tier provides the data services for the application. This architecture makes the application easily manageable and scalable.

The architectural diagram of AVISTED is shown in Figure 3, which provides the description of how the application is organized. The top layer is the Web User Interface, which serves the front-end logic. The front-end logic of the application presents the user with the webpages that allow the user to interact with the web application via a web browser by sending requests from client to server using Hypertext Transfer Protocol (HTTP). It also handles the operations for data visualization using JavaScript libraries such as C3 0 and D3 0. This layer is implemented using HTML5, ASP.NET Razor Pages, and JavaScript. These cutting-edge technologies are supported by most of the modern web browsers.

The next layer is the Application Programming Interface, which provides the business logic for the application. The business logic takes care of the communication between the client requests and the database server. This communication is handled through libraries, functions, objects and variables. The layer is implemented using C# and the ASP.NET Core MVC

Table 2: Functional requirements of AVISTED

ID	Description
R01	AVISTED shall support datasets of different file formats such as ASCII, CSV, HDF5 and NetCDF.
R02	AVISTED shall provide user authentication and authorization.
R03	AVISTED shall support 4 modes: User Mode, Admin Mode, New User Mode, and Guest Mode.
R04	AVISTED shall provide Model Output Management.
R05	AVISTED shall allow the user to upload a dataset of up to 3GB from a single file or multiple files.
R06	The administrator of AVISTED can add a large dataset of size larger than 3GB directly to the repository in the back end.
R07	AVISTED shall allow the user to delete, update and select a dataset.
R08	AVISTED shall provide Model Output featuring the parameters, the date range, and the location details of the selected dataset.
R09	AVISTED shall allow users to select a graph over a specific month and/or year of a variable from the extracted data.
R10	AVISTED shall allow the user to select from a line chart, area chart, bar chart, or scatter chart.
R11	AVISTED shall allow users to compare two or more variables during a specific time period using the charts.
R12	AVISTED shall allow the user to save visualizations in his or her accounts.
R13	AVISTED shall allow users to download the extracted data in ASCII, CSV, HDF5 or NetCDF.
R14	AVISTED shall allow the user to save the extracted data in the user's account.
R15	AVISTED shall allow the user to download or delete the saved archives from his or her account.

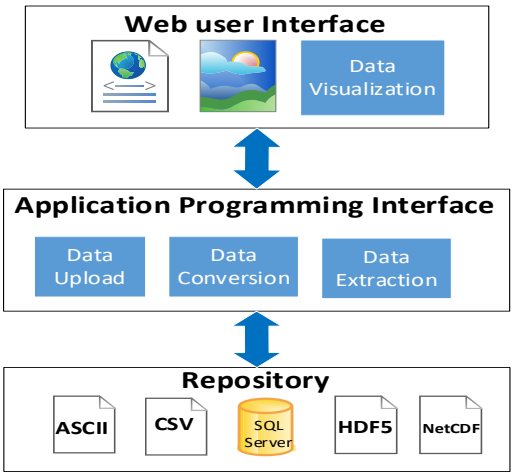


Figure 3: Architectural diagram of AVISTED

Framework. The logic for operations such as data conversion and data extraction is provided in this layer. These functionalities and the statistical computations are implemented using the ASP.NET 4.5 framework. CSV, ASCII, HDF5 data are handled through the RESTful API's and C# libraries such asHDF5DotNet Library 0 and Scientific Dataset Library (SDS) 0 are used for working with the HDF5 and NetCDF files, respectively.

The last layer is the Repository layer, which holds all the data needed for the application. All the files of the large datasets are stored in the repository. These files follow a naming convention that helps the application to retrieve the user-requested data more efficiently. Furthermore, for saving the information related to the users, SQL Server 2016 models and archives have been used.

All the libraries and frameworks used for building AVISTED are open source. The source code of AVISTED application is available in the GitHub repository [2], which makes the application extensible and reusable. Because we used the ASP.Net Core framework and HTML5, the application is platform-independent and supports several devices. Furthermore, we have hosted AVISTED on the IIS server to make it scalable.

4.2 A New Software Engineering Design Notation (GEAD)

The process of building AVISTED involved several phases. After collecting the requirements, we have designed several web user interfaces and drawn UML diagrams [10]. In traditional UML diagrams as shown in Figure 1, there is no link between the GUI designs and the UML activity diagrams. Without this connection it is time consuming for the team to search for the related user interface designs associated with the activity diagrams.

GEAD [1] is a new technique that allows connections between actions in the activity diagrams and their corresponding GUI designs of the user interface. Linking the GUI designs to the actions of the activity diagram allows the software engineering team to check related interface designs and activities quickly. It also helps the developers to identify actions that are common between different GUIs.

If an action has more than one interface design linked to it, it is considered a common task for these designs. Thus, the implementation of such actions can be placed in a global scope for reuse. In an MVC architecture, the common actions can be placed in the same controller and the actions that are specific to a GUI can be placed in the view components. This, in practice, saves a lot of development time.

To illustrate the GEAD approach we present an activity diagram of AVISTED in Figure 4. The small rectangular box at the bottom of each action has a link to the corresponding GUI design of the application. All the actions in the activity diagram have this link thus enabling the team to find the associated GUIs easily. For example, the action Display Model Output Management with User Settings is linked to the interface design #202 shown in Figure 5.

Similarly, the actions Model Output Management, Model

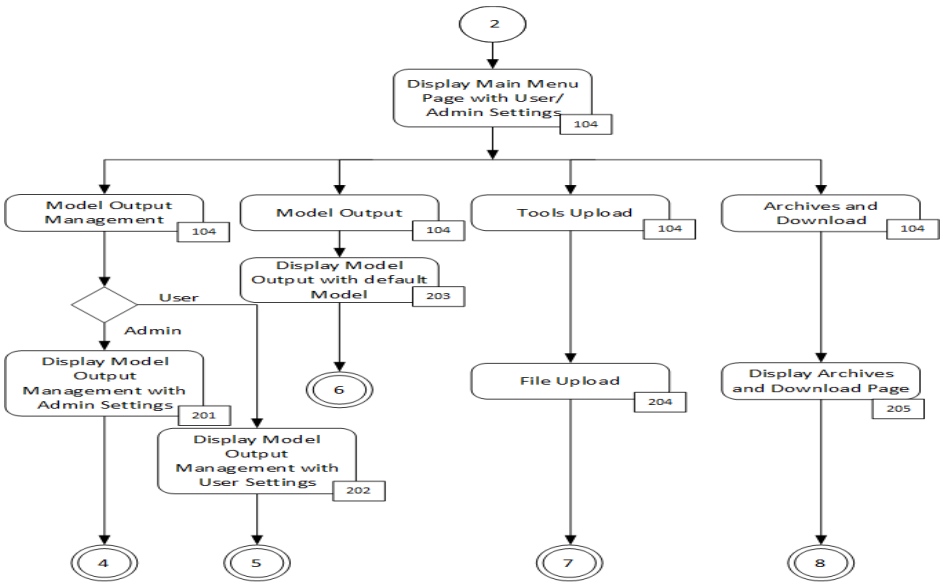


Figure 4: Activity diagram with four modes of AVISTED

Figure 5: Design of model output management page in user mode

Output, Tools Upload, and Archives/Download in the activity diagram are all associated to interface design #104. Since AVISTED is designed using the Model-View-Controller framework 0 the tasks that are related to these activities are placed in the same controller and, respectively, the same view. The actions that are different, for example Display Model Output with Default Model has a different design, #203, so it will be placed in a different controller and view.

This approach provides the developers with a more comprehensive view of the system, so the team can find if there are no designs linked to a significant activity. For example, in a gaming application where the design of the game changes with the state of the game the team can easily identify if more interface designs are needed for a particular state of the game. As GEAD requires interface designs for all major actions the probability of missing interface designs in a software model is much smaller.

In large organizations, GEAD allows the development team to plan the software production more effectively and helps improve the collaboration within the team. For instance, a back-end developer can easily find the front-end developer who is working on the GUI designs that are related to the actions that he or she is working in the back-end. Further, a manager can easily divide the tasks among the team members without any duplication in actions.

5 Application Scenario

In this section, we present an application scenario of a user working with a Nevada regional climate model dataset provided by the NCCP.

The actions required to perform this scenario include selecting a dataset, extracting data, visualizing data and saving it on the local machine. In order to perform the above tasks, the user visits the home page of AVISTED and clicks on the login button. Then the user is required to enter his or her email ID and password on the login page. After a successful login, the user is navigated to the home page, with all the features of AVISTED available, including Model Output Management, Model Output and Archives/Download. For example, the user can select the Model Output Management page, as shown in Figure 6, which provides the user with the list of available models. This feature also allows the users to upload a new model of up to 3GB, view details of existing models, and select a model for further exploration. A model that is uploaded by a user is in the model-under-validation state. This model is checked by the administrator and its state is changed to ready after its configuration and validation are completed.

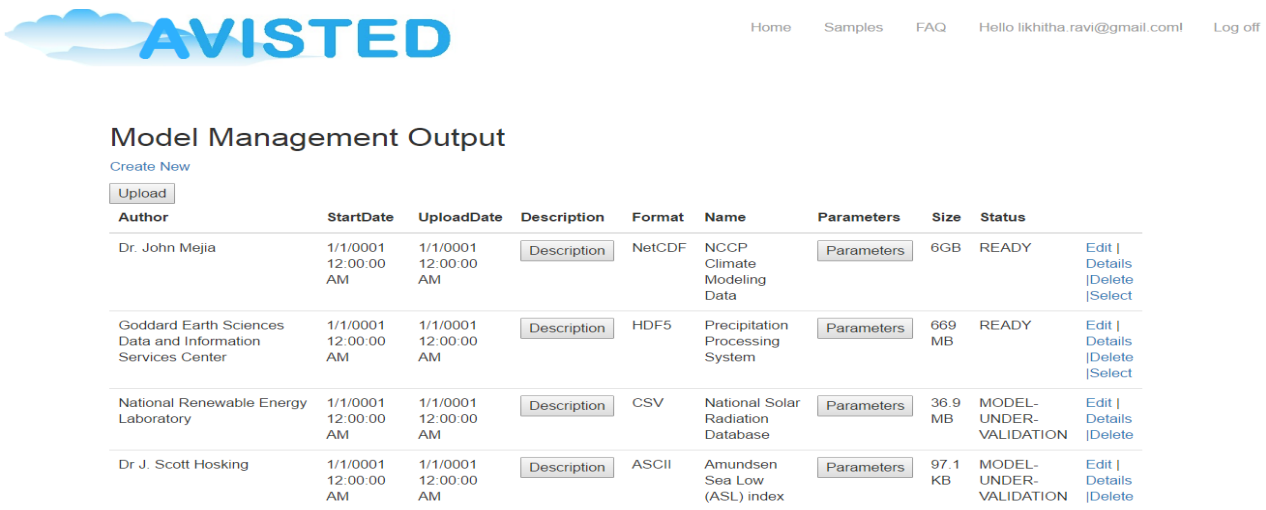


Figure 6: Model management page of AVISTED

Figure 7: Model output page

If the user selects the NCCP climate modeling data, he or she will be redirected to the Model Output page shown in Figure 7. In this figure an NCAR climate model data that has been generated by applying regional climate models upon historical data, which is based on the NCAR re-analysis and CCSM3 is displayed. The size of the NCAR/WRF summary output dataset is about 6 GB. First, climate variables such as precipitation *(precip),* temperature, *(tmin, tmean, tmax),* solar radiation *(qmean, u10mean, vmean),* and snow pack (*swdownmean*) are selected by the user. Then as shown in Figure 7, the user selects a time interval, in this case 10/01/2006 to 12/31/2009. Next, the user selects a location from the polygon area of the map, which indicates that data is available in those regions. Also, the statistics required to be performed on the extracted data must be selected, in this scenario *standard deviation*.

Finally, after selecting the parameters, time range, location, and statistics, the user can either visualize, download, or view the data. In this scenario, the user chooses to visualize the data and

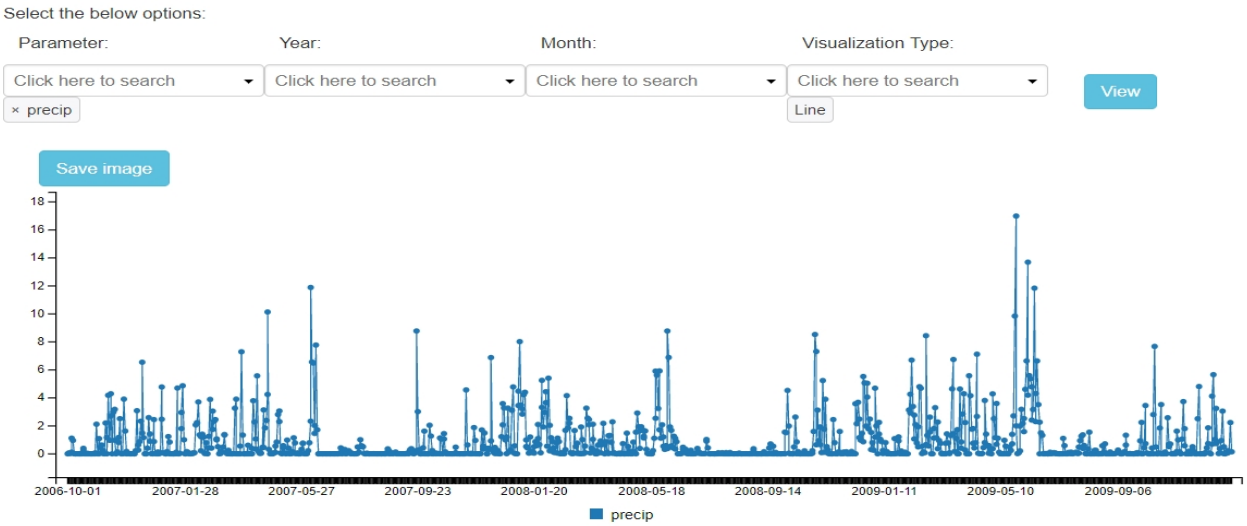


Figure 8: Line chart showing the precipitation parameter

submit the form. Consequently, AVISTED checks the input request and if it is valid the requested data is extracted by making a call to the AVISTED data extractor RESTful API. The API extracts the data as instructed and sends the result back to the web interface. An example of the visualization page is shown in Figure 8 where a line chart of the Precipitation variable is displayed.

AVISTED's visualization tool is flexible and unique. The users can select a combination of different parameters, years or months for time interval, and various types of visualization. AVISTED filters and aggregates the extracted data based on the user's selection of data, time interval, parameters, and visualization type. The resulting data is transformed to JSON format and sent back to the calling JavaScript module. For example, data is aggregated for each year of a parameter when a bar chart is selected.

AVISTED allows users to compare variables, which is a tool very useful for researchers. In the case presented in Figure 9, the user compares the minimum and maximum daily temperatures (*tmin* and *tmax*) in the month of October 2007 using line charts for visualization. The user can also compare several parameters to identify potential trends. To do this, the user selects all parameters of interest and bar chart as visualization type, as shown in Figure 10. Each parameter is aggregated for the selected years.

Finally, the user can find outliers using the scatter plot. For example, the user selects u-radiation (*u10mean*) and v-radiation (*v10mean*) from the parameter list and then chooses a scatter plot for visualization, as shown in Figure 11. Using this plot, an outlier is found in the month of May, 2009 with a very high value for *v10mean*. Similarly, the user can further investigate the dataset by selecting different combinations of parameters and time periods to find the results of the applied model and check his or her hypotheses. Also, the user can save the chart in his or her local machine or on the server under the user's account. To do this, he or she can use the Save Image option provided on the top of each visualization. Once the image is by using the Archives and Download feature of AVISTED.

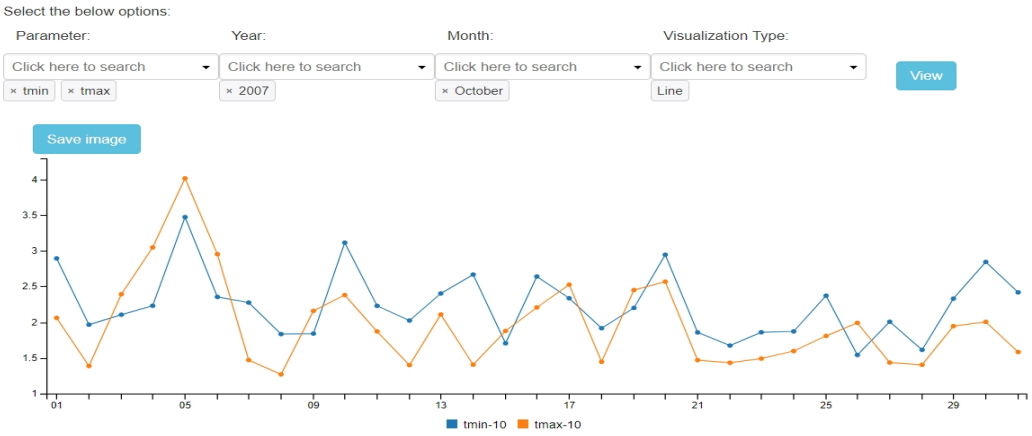


Figure 9: Line chart with min and max temperatures in the month of October 2007

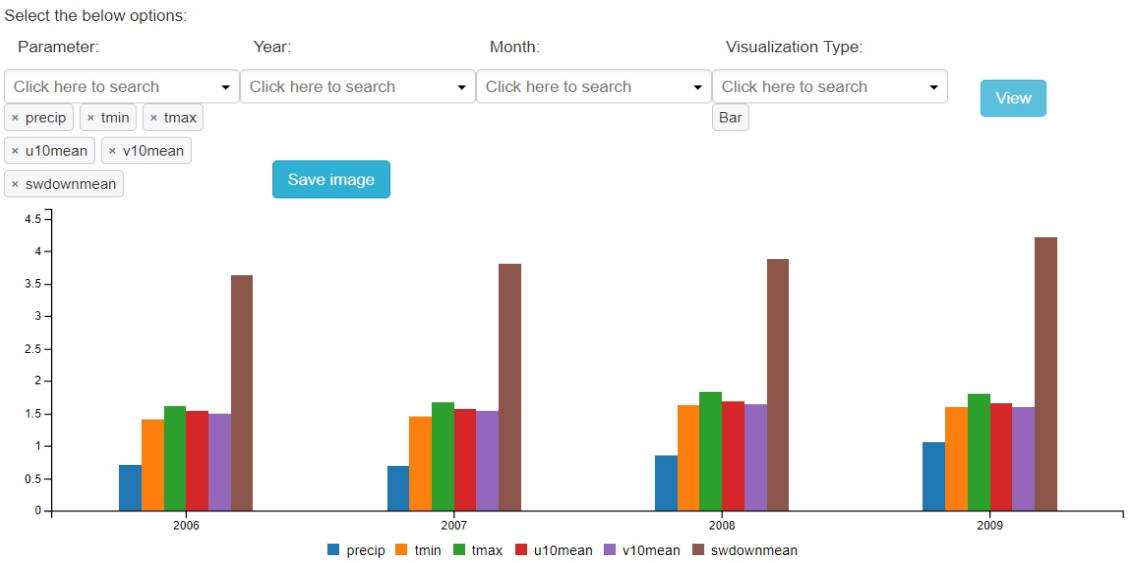


Figure 10: Bar chart with all the variables

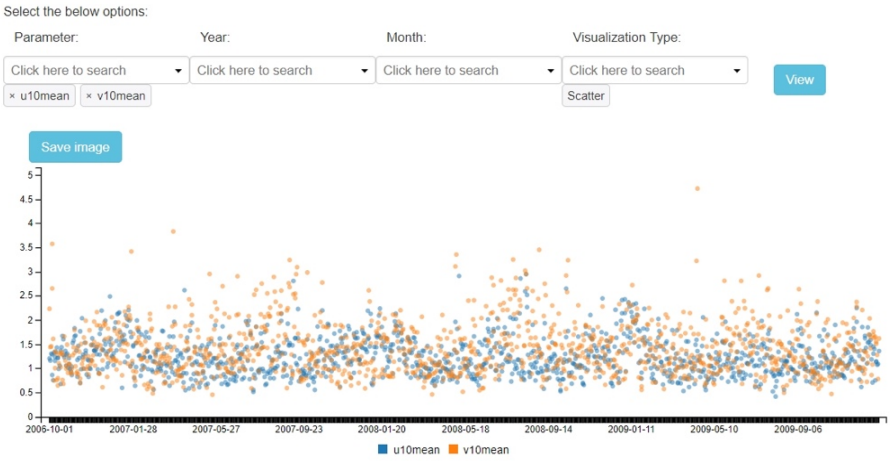


Figure 11: Scatter plot with the u10mean and v10mean variables from the NCAR dataset

6 Comparison with Related Work and Discussion

6.1 AVISTED Related Work

Certainly, there exist some works related to AVISTED, for example Cal-Adapt [9], Plotly [21], Climate Explorer [5] and DPT-VW 0 are used for data visualization and processing. These are described in some more detail next.

Cal-Adapt [9] is a web application that provides climate change information for California. The application presents research work done by California state agencies. Many datasets with climate projections for climate variables, such as snow pack, wildfires, flooding, and temperatures, from 1950 through 2099 are available on the Cal-Adapt website. The users can use the tools available to visualize the data for each model as well as compare the values of climate variables between the models. The users can also select a location on the map. The tool also allows users to calculate the statistics based on historical annual mean or modeled projected annual mean. It provides many scenarios and uses different models for predicting the climate change data of California. These functionalities make the toolset more unique and sophisticated than many others, but Cal-Adapt's impressive features would have been even more beneficial if it was an open source project. As of now, it cannot be reused or extended by other communities who are doing similar research. Also, some functionalities such as data upload and editing are not supported by its toolset.

Plotly [21] is a generic web-based data visualization toolset with a collection of high-quality complex visualizations and dashboards. It provides free open-source libraries in JavaScript, Matlab, Python, and R languages for further customization of the plots. The users can upload data from a single or from multiple CSV files. If multiple files are uploaded, each file is placed in a separate grid. Once the data is ready, charts can be created by selecting the grid and chart type. A unique feature of Plotly is that it allows users to collaborate with others who

would like to view and edit the plots on their work through an email with link to the generated graph. Although the features of Plotly are very promising for users, it is not very suitable for large datasets. Another limitation of Plotly is that it allows data upload only in CSV and TSV formats.

Climate Explorer [5] is a tool developed by Habitat Seven [11] and hosted by NOAA's National Centers for Environmental Information. The tool allows users to explore historical and projected climate change data for all counties in the US. Users can use the map to view the data for all the counties or select a county by entering the county name, city, state, or zip code. After selecting a county, users can explore that area or select other nearby weather stations using the map. Temperature and precipitation information for the county is provided. In spite of all these very useful features, the usage of the tool is somewhat limited because it does not allow data upload or data extraction, and the graphs provided by the tool are not interactive. Users are only provided with a line chart or a map. The application is not open source, and there is no API for customizing or extending the visualizations.

Data Processing Toolset for the Virtual Watershed (DPT-VW) is another tool designed and implemented by our group to process environmental datasets 0. The toolset is fairly comprehensive and some ideas used in its design have also been used in AVISTED. However, the DPT-VW toolset has limitations. For example, it does not support some useful file format conversions, such as to ASCII and HDF5. Also, the visualization component is not very powerful. For example, the users cannot select a particular month or year for visualization. Additionally, DPT-VW allows only a single data file upload, which is a limitation if the user is interested in data contained in multiple files. These aspects of DPT-VW were improved in AVISTED.

6.2 GEAD as a New Software Engineering Design Notation

At present, there are many tools available for designing software applications. They can be classified into categories based on their features like general purpose, special purpose, specific language, code generation, executable UML, desktop application, mobile application, online application, open source, commercial and freeware.

The following are some of the current prevalent UML drawing tools: Rational Rose, SmartDraw, Microsoft Visio, LucidDraw, and ArgoUML. Some of these have support for activity diagrams and GUI builders, some have for building GUIs, and some have support for both activity diagrams and creating GUIs, like Visio, but the latter are static. None of them provide links between activity diagrams and GUI, which is an original idea of this paper's authors. These links are possible by building an HMTL-based UML tool that allows drawing UML diagrams, designing GUIs, and providing HTML links between them. A similar HTML web application that allows drawing UML diagrams online is Diagramo [15]. Specifically, it allows building UML designs online but it does not support links. Tools such as Dreamweaver, Sublime, Coda 2, and Brackets 0 are used in building such web applications. Tenzer 0 developed an interactive game to improve the UML designs. The user of the game is allowed to explore several variations of the designs while playing the game. This allows the users to improve their designs interactively.

6.3 Discussion

After comparing AVISTED with similar tools, we conclude that AVISTED fills the gap between general purpose and climate-related data visualization tools. The strength of AVISTED lies in its support for datasets which are in climate data formats. Also, the flexibility it offers to users in selecting a dataset, data variables, statistics, conversion, and visualizations is unique in climatology. Moreover, creating user accounts and giving users a choice of saving their visualizations and data selections is also not observed in other tools in this field.

On the other hand, AVISTED does not allow users to edit a dataset after uploading; with the large amount of datasets it supports, this will be a very complex task. Furthermore, at present the toolset does not support features such as editing visualizations, sharing visualizations on social media, or embedding them into another web application. These features, however, can be relatively easy to add to the toolset with some add-ons to the client-side JavaScript functions in the future. More details about a comparison of AVISTED with similar tools can be found in the thesis completed in 2018 by the first author of this paper [22].

7 Future Work and Conclusion

In this paper, we have introduced the new software application AVISTED, which has four main components: data upload, data extraction, data visualization, and data conversion. The software requirements and design models of AVISTED were described together with its main capabilities, an application scenario has been presented, and a comparison with related work and a discussion have been provided. We have also introduced the new GEAD software design approach and illustrated it as we applied it to AVISTED's development.

In the future, we would like to add several other useful features and functions to AVISTED, such as data processing using executable environmental models and comparisons of their results. We also intend to optimize the visualization component of AVISTED. Some preliminary work has been done and presented in our recent papers 0, 0. Also, we plan to support other data formats and aim to add more datasets to AVISTED from other repositories. Furthermore, we would like to improve the usability of the application by involving the climate researchers, getting their feedback, and incorporating it in a future version of the application. In addition, more testing shall be conducted to evaluate the performance of the application when the number of its concurrent users increases. Lastly, we plan to work on the development of a supporting tool

for GEAD and refine the GEAD technique by using the new tool in software-intensive development projects.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant numbers IIA-1329469 and IIA-1301726. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] "ASP.NET MVC Overview," available at: https://msdn.microsoft.com/en-us/library/dd381412 (v=vs.108).aspx, [Accessed November 4, 2017].

[2] "AVISTED," available at: https://github.com/likhitharavi/ AVISTED/tree/master/A.VISTED, [Accessed August 20, 2018].

[3] C3.js| "D3-Based Reusable Library", available at: http://c3js.org/, [Accessed November 4, 2017].

[4] "Choosing the Right Text Editor | Brackets, Sublime Text, Coda, and More," available at: http://blog.digitaltutors, com/brackets-coda-sublime-text-text-editor-choose/, [Accessed November 4, 2017].

[5] "Climate Explorer,", available at: https://toolkit.climate, gov/climate-explorer2/, NOAA [Accessed 28 June 2017].

[6] "Climate Nexus," available at: https://climatenexus.org/, [Accessed June 25, 2018].

[7] D3.js – "Data-Driven Documents,", available at: https://d3js.org/, [Accessed November 4, 2017].

[8] "DataONE," available at: ww.dataone.org, [Accessed June 25, 2108].

[9] M. S. Deas, "Cal-Adapt and the Usability of Climate Adaptation Tools," Massachusetts Institute of Technology, 2015.

[10] M. Dumas, and A. H. Ter Hofstede, "UML Activity Diagrams as a Workflow Specification Language," In UML 2185:76-90, October, 2001.

[11] "Habitat Seven," Available: http://habitatseven.com/, [Accessed 28 June 2017].

[12] "HDF5DotNet,", available at: http://hdf5.net/, [Accessed November 4, 2017].

[13] M. Hossain, H. Munoz, R. Wu, E. Fritzinger, S. M. Dascalu, and F. C. Harris, "Becoming DataONE Tier-4 Member Node:Steps Taken by the Nevada Research Data Center", *Proceedings of Optimization of Electrical & Electronic Equipment Aegean Conference on Electrical Machines & Power Electronics (OPTIM-ACEMP 2017)*, Brasov, Romania, pp. 1089-1094, May 25-27, 2017.

[14] M. Hossain, R. Wu, J. T. Painumkal, M. Kettouch, C. Luca, S. M. Dascalu, and F. C. Harris Jr,. "Web-Service Framework For Environmental Models," *Proceedings of the IEEE Conference on Internet Technologies & Applications 2017 (ITA 2017)*, Wrexham, North Wales, UK, 6 pp, September 12-15, 2017.

[15] "HTML5 Diagram Editor," available at: http://diagramo, com/editor/editor.php, [Accessed November 4, 2017].

[16] "Modeling Output," available at: http://sensor.ne vada.edu/NCCP/ Downloads/ Modeling%20Output.aspx, [Accessed November 4, 2017].

[17] National Oceanic and Atmospheric Administration, U.S. Department of Commerce, available at: http://www.noaa.gov/, [Accessed 8 December 2017].

[18] Nevada Research Data Center, available at: http://www.sensor.nevada.edu /NRDC/, [Accessed November 4, 2017].

[19] "Nevada System Sponsored Programs and EPSCoR," available at: https://epscorspo.nevada.edu/, [Accessed November 4, 2017].

[20] L. Palathingal, R. Wu, R. Belkhatir, S. M. Dascalu, and F. C. Harris, "Data Processing Toolset for the Virtual Watershed," *Proceedings of the 2016 International Conference on Collaboration Technologies and Systems (CTS 2016)*, Orlando, FL, pp. 281-287, October 31-November 4, 2016.

[21] "Plotly,", available at: https://plot.ly/dashboards-and-reports/, [Accessed December 6, 2017].

[22] L. Ravi, *AVISTED: Analysis and Visualization Toolset for Environmental Data*, PhD dissertation, University of Nevada, Reno, 2018.

[23] L. Ravi, S. Dascalu, and F. C. Harris, "GUI-Enhanced Activity Diagrams with Application to the Design of AVISTED," *Proceedings of the 24th International Conference on Software Engineering and Data Engineering (SEDE-2015)*, pp. 436-442, October 2015.

[24] L. Ravi, S. Dascalu, F. C. Harris, J. Mejia, and N. Belkhatir, "VISTED: A Visualization Toolset for Environmental Data," *Proceedings of the 2015 International Conference on Computers and Their Application (CATA-2015)*, pp. 335-342, March, 2015.

[25] Scientific Data-Set Library, available at: https://www.microsoft.com/en-us/download/details.aspx ?id=52412, [Accessed November 4, 2017].

[26] J. Tenzer, "Improving UML Design Tools by Formal Games," *IEEE Explore Proceedings of the 26th International Conference on Software Engineering, (ICSE-2004)*, 2004.

[27] R. Wu, C. Chen, S. Ahmad, J. Volk, C. Luca, F. Harris, and S. Dascalu, "A Real-time Web-based Wildfire Simulation System," *Proceedings of the 2016 IEEE Industrial Electronics Conference (IECON 2016)*, Florence, Italy, pp. 4964-4969, Oct 24-27, 2016.

[28] R. Wu, S. Dascalu, and F. Harris, "Environment for Datasets Processing and Visualization Using SciDB," *Proceedings of the 24th International Conference on Software Engineering and Data Engineering (SEDE 2015)*, San Diego, CA, pp. 223-229, October 12-14, 2015.

[29] R. Wu, J. Painumkal, S. M. Dascalu, and F. C. Harris Jr, "Self-managed Elastic Scale Hybrid Server Using Budget Input and User Feedback," *Proceedings of the 12th Workshop on Feedback Computing, as part of the*

Proceedings of the 14th International Conference on Autonomous Computing, , Columbus, Ohio, 6 pp., July 17-21, 2017.

[30] R. Wu, J. T. Painumkal, N. Randhawa, L. Palathingal, S. R. Hiibel, S. M. Dascalu, and F. C. Harris, "A New Workflow to Interact with and Visualize Big Data for Web Applications," *Proceedings of the 2016 International Conference on Collaboration Technologies and Systems (CTS 2016)*, Orlando, FL, pp. 302-309, October 31-November 4, 2016.

[31] R. Wu, J. T. Painumkal, J. M. Volk, S. Liu, S. J Louis, S. Tyler, S. M. Dascalu, and F. C. Harris, "Parameter Estimation of Nonlinear Nitrate Prediction Model Using Genetic Algorithm," *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2017)*, San Sebastian, Spain, pp. 1893-1899, June 5-8, 2017.

Likhitha Ravi received in 2018 a PhD degree in Computer Science and Engineering from the University of Nevada, Reno, USA. She also received in 2007 a Master's degree in Computer Engineering from Texas A&M University, Kingsville, USA and in 2005 a Bachelor's degree in Computer Science and Information Technology from Jawaharlal Nehru Technology University, Hyderabad, India. Her main research interests are in the field of data analysis and data visualization over the web.

Eric R. Fritzinger received in 2003 a BS and in 2006 an MS, both in Computer Science, from the University of Nevada, Reno. After spending several years in the field of medical robotics, he returned to UNR to participate in a state-wide project studying the effects of climate change in the Great Basin, Nevada. He has worked on model and data interoperability as well as management and organization of environmental data. He is currently the lead developer for the Nevada Research Data Center, based out of UNR's Computer Science and Engineering Department.

Department of Computer Science and Engineering at the University of Nevada, Reno, USA, which he joined in 2002. He received in 1982 a Master's degree in Automatic Control and Computers from the Polytechnic University of Bucharest, Romania, and in 2001 a PhD in Computer Science from Dalhousie University, Canada. His main research interests are in the areas of software engineering and human-computer interaction. He has published over 180 peer reviewed papers and has been involved in projects funded by industrial companies as well as federal agencies such as NSF, NASA, and ONR.

Frederick C. Harris, Jr. is currently a Professor in the Department of Computer Science and Engineering and the Director of the High-Performance Computation and Visualization Lab and the Brain Computation Lab at the University of Nevada, Reno, USA. He received his BS and MS in Mathematics and Educational Administration from Bob Jones University in 1986 and 1988 respectively, and his MS and PhD in Computer Science from Clemson University in 1991 and 1994, respectively. He is a member of ACM (Senior Member), IEEE, and ISCA (Senior Member). His research interests are in parallel computation, computational neuroscience, computer graphics, and virtual reality.

A Review on Deployment of Algorithms for Cloud Internet of Things Application Domains

Edje E. Abel*
Universiti Teknologi (UTM), Johor Bahru, MALAYSIA
Delta State University, Abraka, NIGERIA

Muhammad Shafie Abd Latiff†
Universiti Teknologi (UTM), Johor Bahru, MALAYSIA

Abstract

Cloud Internet of Things (IoT) is an emerging technology that is already impelling the daily activities of our lives. Its utilization is projected to make the computer and network communication hardware components, Things (e.g. sensor nodes, actuators, and RFID tags), protocols and software invisible to the public while in operation. Thus, making this new paradigm a significant component of the Future Internet. However, the enormous resources (data and physical features of Things) generated from CloudIoT, are lacking suitable managerial approaches. Existing research work in literature has surveyed CloudIoT based on its fundamentals, definitions and layered architecture as well as security challenges. However, to the best of our knowledge, none of the existing researches are yet to provide a detailed analysis on the approaches deployed to manage the heterogeneous and dynamic resources generated by sensor devices in the CloudIoT paradigm. Hence, to bridge this gap, we investigated and analyzed the existing algorithms designed to manage the aforementioned resources deployed in the various CloudIoT application domain. In this article, we present the emergence of CloudIoT, followed by previous related survey articles in this field, which motivated the current study. Furthermore, the utilization of simulation environment, highlighting the programming languages and a brief description of the simulation packages adopted, to design and evaluate the performance of the algorithms are examined. The utilization of diverse network communication protocols and gateways to aid resource dissemination in the CloudIoT network infrastructure are also discussed. The future work as discussed in previous researches, which pave the way for future research directions in this field is also presented, and end with concluding remarks.

Key Words: Internet of things sensing devices, radio frequency identification, network communication protocols and gateways, and cloud platform.

* Faculty of Computing, Department of Computer Science and Faculty of Science, Department of Computer Science. Email: toboredje@gmail.com
† Faculty of Computing, Department of Computer Science. Email: shafie@utm.my.

1 Introduction

The Internet of Things (IoT) model is established on intelligent and autonomous-configuring smart devices, interconnected in real time basis to form a worldwide network infrastructure with the support of the internet. It signifies one of the most sensational technologies, enabling cybernetic and pervasive computing developments. IoT is largely categorized as physical and embedded small things (such as sensor nodes, actuators, and RFID tags) that are extensively distributed, constrained with a limited energy power source, computational processing power, and storage capability. Furthermore, issues such as performance and reliability based on the quality of service and experience (QoE) delivery are a major concern in IoT development. On the other hand, cloud computing has virtually unconstrained capabilities in regards to computational processing power and storage, is considered a more established paradigm, and have most of the IoT challenges resolved. Thus, an innovative information technological infrastructure in which cloud and IoT are two complementary technologies merged together is expected to revolutionize both present and Future Internet, called CloudIoT [12, 17, 83]. Investigating the comprehensive and coherent previous research in this field, it was discovered that the topic (CloudIoT) has attracted more acknowledgment over the last seven years. Inspired by previously related research, we investigate the literature focusing on the deployment of algorithms, used for the management of both data resource and the physical features of Things (e.g. sensor nodes and RFID tags) in the various CloudIoT application domain. To this end, our contributions to this research work are as follows;

- We analyze the algorithmic processes designed for managing the data resources and the physical features of the IoT sensing device(s) deployed in the different CloudIoT application domain.
- We present the CloudIoT simulation environment, highlighting the programming languages and briefly discussing the simulator software packages, deployed for the design and performance evaluation of the existing algorithms.
- A comprehensive discussion on the physical components

(network communication protocols, sensing devices, cloud data center, and gateways) for the realization of the CloudIoT application platforms.
- An extensible discussion on future work as stipulated in previous researches which pave the way for future research directions in this field.

The major benefit of this research is to assist prospective academic researchers to understand the current status of deploying algorithms in CloudIoT, as well as critical gaps to address in the future. Hence, to conserve the energy of connected IoT sensing devices and optimal usage of the cloud features. Consequently, for the provisioning of effective, reliable and efficient services to end users. A systematic research procedure is adopted as depicted in Figure1 below. Thus, to formulate the systematic stages that constitute the research under study, to establish a qualitative approach of the chronological events leading to the actualization of the research goal. Thus, build the basis for the structure of this article as follows.

A detailed discussion on the emergence of CloudIoT (Section 2) is followed by a brief analysis of previous related research in this field (Section 3) which motivated the current study. Furthermore, a detailed research methodology is presented to explain how the research under study was conducted (Section 4). Followed by a compressive analysis of algorithmic (techniques) processes deployed to manage the huge dynamic data resource(s) and the physical features of IoT devices in the various CloudIoT application domain (Section 5). The presentation of CloudIoT simulation environment (Section 6) and CloudIoT physical components such as the network communication protocols and gateways (Section 7) are discussed extensively. Next, the future research directions (Section 8) are also presented in this article and end with concluding remarks.

2 The Emergence of Cloud Internet of Things

The emergence of cloud and the internet of things tend to virtualize the computer technological industries. An era where computing hardware, software, and network capabilities are rendered virtually as services to end users. The founders of the MIT Auto-ID Center formulated the phrase "Internet of Things" two decades ago. MIT Auto-ID Center was co-headed by Kevin Ashton in 1999 and David L. Brock in 2001. Sundmaeker, et al. in [70] defined Auto-ID as any wide-range session of identification devices used in the industry to automate, reduce errors, and enhance efficiency. Therefore, devices such as barcodes, biometrics smart cards, voice recognition, sensors, actuators and Radio Frequency Identification (RFID) are classified as Auto-ID devices. RFID has been the leading Auto-ID technology deployed in the automation industries since 2003. It is an automated data collection device that utilizes radio frequency signals to transfer data between a reader and a movable object to identify, categorize and track that object [19]. For instance, objects such as raw materials and finished products can be tracked and identified from the production stage to the distribution center as well as in the shop outlets. Furthermore,

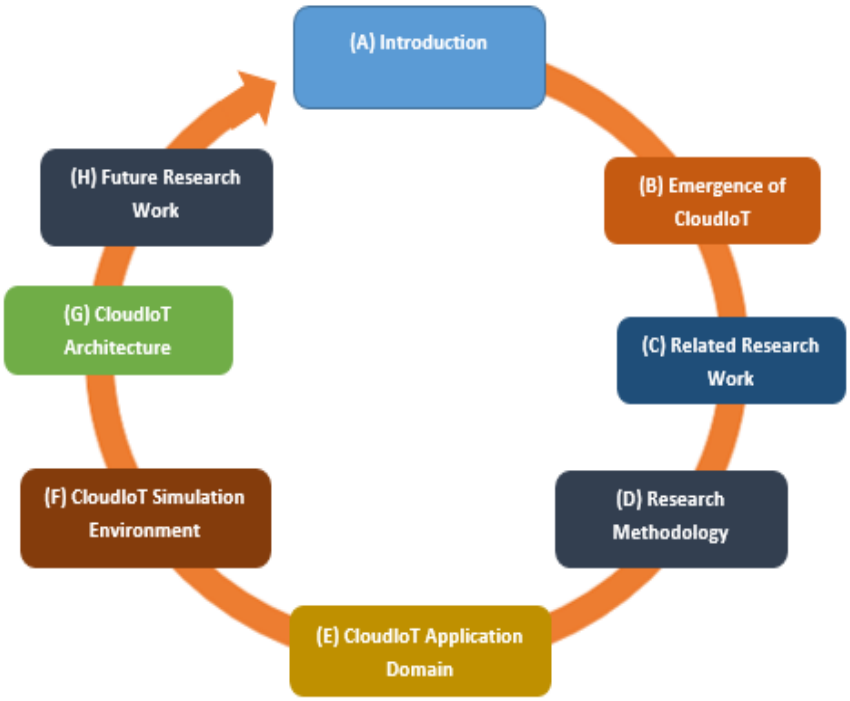


Figure 1: The research structure

it is commonly used by courier service and manufacturing industries for monitoring and identifying of products in transits in real time basis.

Kevin Ashton stated that the "IoT is considered then as the mere extension of Radio Frequency Identification as it is a form of amoeba of the wireless computing world" [71]. Hence, a new era of technology has emerged in the world of computing. In the nineteenth century, machines learned to work, they learned to think in the twentieth century and to comprehend in the twenty-first century. Internet of Things came to limelight in the year 2005 when the International Telecommunications Union initially published the report on the subject. They suggested that the Internet of Things tends to connect objects sensory and intelligent manner by feeling things, thinking things, shrinking things and tagging of things with the aid of combining RFID, IoT sensor device(s) and nanotechnology. Statistics show that there were about 1.5 billion internet-based connected things such as cell phones in the year 2010. Presently, the rapid growth in the development and adoption of IoT sensor device(s) due to its portability have increased the number of internets connected things to an estimated 7.5 billion. IoT sensor device(s) are tiny low powered chips deployed to link the physical with the digital world, by capturing and revealing real-world activity that can be stored, processed and act upon in real time basis (Dargie and Poellabauer, [20]). It has the ability to communicate its readings to the internet cloud services for further processes and trend analysis [69]. On the other hand, cloud computing is a model of service delivery and access where dynamically scalable and virtualized resources are provided as a service over the Internet [63]. The origin of cloud computing dated back in the year 1959, when John Macarthy envisaged that it is necessary to introduce time-shared-computers. Specifically, for the sharing of computing resources such as software and hardware between two or more clients with the support of multi-tasking and programming simultaneously. In the year 1961, he buttresses the opinion that computers should become a utility like telephone services, which motivated Douglas Parkhill to base his research on the feasibility of utility computing in the year 1966. His research results show that the propensity of actualizing a utility computing service is unrealistic. However, his research outcome was shattered when Amazon started selling books via the internet in the year 1997 [43]. In 1999 Salesforce.com embarked on the provisioning of software as service on the internet which is accessible on a payment basis. Furthermore, Google joined the trends in the year 2004, by providing free email services that enable clients to send messages to their loved ones across the globe. Thus, extending their services by launching a public cloud called Google Cloud Platform (GCP). It comprises a collection services such as storage space, application, machine learning, big data, artificial intelligence and the internet of things, which are rendered to prospective clients (e.g. software developers, cloud administrators, and IT professionals) and the general public. These services can be accessed from Google platform via a dedicated network connection or the public internet. The main cloud computing services in GCP are as follows: Google Compute Engine which allows clients (end users) to have access to virtual machines for the hosting of workload, Google Application Engine (SaaS), which enable software developers to have access to scalable toolkits that runs on the GCP, for the development of software products, Google Cloud Storage (IaaS), which provides huge storage space for both structured (MySQL) and unstructured (NoSQL non-relational) data sets and the Google Container Engine, that is utilized to stage-manage Ducker containers running within the GCP. Similarly, Elastic Cloud computing (EC2) that enable numerous end users to pay and make use of applications dynamically on-demand basis over the internet was launched by Amazon [15]. After which, Amazon web services (AWS) was introduced in the year 2006 to complement the services of EC2 by providing storage hosting (S3) and platform for software developers (PaaS) on a pay as you go basis. Presently, Amazon offers over ninety services via its AWS platform which includes Internet of Things facilities, networking, databases, data analysis tools, and mobile developer toolkits etc. Additionally, it renders a huge volume of virtual servers at a cheaper rate than the physical servers to subscribers. AWS can be accessed over Hyper Text Transfer Protocol (HTTP), by utilizing the Representational State Transfer (REST) method and Simple Object Access Protocol (SOAP) protocol.

Microsoft introduced a cloud service called Azure, which provides services to end users such as hardware storage space, applications and website hosting in the year 2010. Presently, cloud computing services are highly in demand as business organizations and other multi-national institutions rely heavily on its services ranging from Software as a Service (SaaS), Platform as a service (PaaS) and Infrastructure as a Service (IaaS). According to the reports published by [60] and [78] in the year 2017, stipulated that AWS controls and provides over 30% of Cloud's Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) globally, as compared to its three major business rival Google (8%), Microsoft (11%) and IBM (6%) respectively.

Over the last six years, research scholars and industrialists are working towards the integration of Cloud and IoT technology as both tend to complement each other. IoTs is known for generating an enormous amount of data resources and is constrained by processing power, storage space, and energy. On the other hand, cloud computing can offer an effective solution to leverage the constrained IoTs by providing its unlimited virtual features and resources such as storage space and high computational processing power. Thus, "extending the scope of IoTs to deal with real-world things in a more disseminated manner, for the provisioning of new services in real-life scenarios" [11], therefore, leading to the formulation of a new era in the Computer Information Technological Industry, called Cloud Internet of Things. The new service paradigm includes Sensing as a Service (SaaS), Sensor Event as a Service (SEaaS), Sensor as a Service (SaaS), Ethernet as a Service (EaaS) and Video Surveillance as a Service (VSaaS). SaaS allows virtual access to manage remote sensors and enabling the provisioning of sensor data. SEaaS allows the dissemination of messaging services obtained from sensor event coverage, while EaaS enable virtual layer-2 connectivity to remote devices [82].

VSaaS provides ubiquitous access to recorded video and implementing complex analyses in the Cloud [56].

3 Related Research Work

Jing et al. in [31] presented a survey on the security challenges of the Internet of Things (IoT) technology by analyzing the security issues of each of the IoT layers, which includes perception layer, transportation layer, and application layer. They also analyze the cross-layer heterogeneous integration issues in detail and tried to find a solution to resolve the whole problem. Furthermore, comparing the security between the conventional network and IoT as well as discussing future research directions in IoT security issues. An overview of IoT, highlighting some enabling technologies, protocols and application challenges is presented by [4]. Thus, enabling application developers and researchers to efficiently merge diverse protocols together in order to satisfy desired functionalities. The relationship between the IoT and other emerging technologies such as big data analytics, cloud, and fog computing are highlighted and presenting a service use-case to demonstrate the integration of various protocols for the provisioning of desired IoT services. Li et al. in [40] carried out a survey on the definitions, architecture, fundamental technologies and applications of IoTs. Hence, numerous definitions from related articles and the emerging techniques used for the implementation of IoTs are systematically analyzed. Also, some open challenges with respect to IoT applications are explored. Peinl et al. [54] carried out a survey on the management of Docker cluster in the cloud, which only manages containers on one host. They analyze how Sensor as a Service (SaaS) platform enables a container to manage a solution for multiple hosts and identifying gaps and integration requirements, as well as formulating integration components with enhancement tools to close the gaps.

A survey on the communication features for interaction with languages, protocols, and architecture that enforces today's standard and software developments of cloud technology is presented [37]. It envisages that attention will be focused toward multiplexing and encryption in an upcoming transport mechanism in the future. Botta et al. [12] mainly focuses on the integration of cloud IoT paradigm. They presented the driving force currently behind the integration of both technologies, the current applications and the challenges of merging them. Also, they were able to resolve some of these challenges with the enhanced formulated integration components and tools. Cavalcante et al. [14] in their research, reviews a detailed and comprehensive understanding of the integration of the internet of IoT and cloud computing, highlighting the current state of research in cloud IoT and identifying important gaps in the existing approaches as well as future research directions.

Aitsaadi et al. [3], addresses the issues of using the cloud infrastructure with the aid of IoT systems from the sensors and machines to end users, as well as applications hosted on the cloud. It also analyses the deployment efforts on IoT infrastructures, gateways, and cloud platform. Ngu et al. [50] embark on a thorough survey on the capabilities of existing IoTs middleware. This research was motivated by the need for an IoT middleware application designed for timely estimation of blood alcohol content using smartwatch sensor data. In addition, challenges and other enabling technologies that allow the heterogeneity of IoT devices and its adaptability were discussed. Consequently, the security issues involved in the development of IoT middleware are also presented. Ray [62] carried out a research survey on IoT cloud platforms in order to resolve various service management domains, such as device management, heterogeneity, data management, application development, visualization and tools for analysis. In addition, IoT cloud service providers in regard to their expertise and weaknesses, in reality, are also discussed.

Tayeb et al. [72] embarks on the review of a high-level conceptual layered architecture for IoTs, cloud computing and fog from a computational viewpoint. The architecture emphasizes issues related to cloud computing, fog, sensors, and actuators, as well as how they interconnected by communication network infrastructures. Hence, none of the aforementioned review or research surveys are yet to consider the utilization of algorithms, deployed for the management of things resources (data and physical features) in the cloud-enabled IoT paradigm. Thus, motivated the research study currently under study.

4 Research Methodology

The research methodology adopted to conduct this research review has been used previously by Gonzales-Martinez et al. [26]. The literature exploration covered contributions from the year 2012 to 2017 and was performed utilizing the databases deliberated to be suitable to discover the targeted studies: Springer, Scopus (Elsevier), MDPI and IEEE Xplorer Digital Library. The search phrase used was: (“cloud internet of things” OR “internet of things application”) and (“cloud software as a service” OR “internet of things”). This search phrase was considered with the intention of obtaining a numerous number of the studies available in the databases that were relevant for the review despite the outcome of the query returning many other works that are not related to the targeted studies. Thus, studies or literature published in the English language and contained in journals (both printed and electronic), white papers, conference proceedings, and books were only considered. The initial search results obtained is about 256 candidate articles. Each candidate article undergoes thorough stages of assessment (a. assess the title and reject the ones that unrelated to cloud internet of things applications, b. reading the abstract carefully and discard the irrelevant ones, c. retrieve the study and meditate on the introduction and conclusion, reject if the contribution is similar to other more related article or study by the author and d. discard the article with low quality after painstakingly assessing its quality of contribution) before it was finally selected. Consequently, the quality parameters such as the significance of the contribution for the cloud of things applications, the level of relevance of the study with a cloud of things application domain as well as the novelty, flawless and the writing quality of contributions were considered.

A total of 44 studies passed the quality assessment stage.

Figure 2 below denotes some of the bibliometric data of the 44 studies of the selected article. After which the extraction of data process was piloted to retrieve the following information from each contribution; types of application domain in cloud internet of things infrastructure, research questions, algorithm/techniques deployed to address research challenges, the models/network communicating channels in cloud of things application platforms, programming languages utilized for the development as well as the simulation packages deployed to determine the validity of its results and the gateways deployed in the cloud of things infrastructure. Thereafter, a qualitative analysis was performed on the 44 selected articles to synthesize the actual findings in regards to the objective of the current study. Thus, to identify the types of cloud internet of things application domains (health, manufacturing, agriculture and environmental), the algorithmic (techniques) processes deployed to resolve the research challenges, the simulation package utilized to determine the performance of the CloudIoT application, so as to validate its result, the various network communication channels/gateways deployed between the cloud and internet of things sensing devices for realizing application requests and the identification of future works as stated in the selected articles, which pave the way for future research areas.

5 Cloudiot Application Domain

CloudIoT infrastructure has been applied to various institution domains and the environment for monitoring and the provisioning of timely information that will aid mankind, to predict and avoid unpleasant circumstances before they occur. These institutional domains include the medical health institution, manufacturing, agriculture and smart environment, as discussed as follows.

5.1 Medical Health Domain

In the medical health, legacy and modern healthcare facilities are tagged with RFIDs and wireless sensor nodes for safety purposes. Information retrieved from the IoT device(s) are transmitted to the cloud for onward computational process and analysis. Thus, to ascertain the conditional status of the healthcare facilities, which can only be accessed by authorized healthcare personnel, via an internet connected device such as PCs, laptops and mobile phones. Consequently, wearable body-sensor nodes (WBN) are worn to monitor the health status and facilitate treatments to patients with critical health issues in a real time basis. Figure 3 depicts a typical CloudIoT ecosystem deployed in the medical healthcare domain.

Diallo et al. [21] propose a real query processing optimization-time algorithm for cloud-enabled Wireless Body Area Sensor Networks (WBANs). The proposed algorithm is based on the polling-based communication protocol, utilized to minimize the data transmission latency from sensor nodes in the WBANs to the cloud platform. Energy is conserved by adopting the Convex Cost algorithm, which determines the optimal values of polling intervals for each of the WBAN to obtain the globally minimum cost. Furthermore, it computes the error tolerance and the reliability interval on the patient sensed data stored on the medical database server, residing in the cloud platform. Mendes et al. [46] proposes the Cross-layer Dynamic Admission Control for Cloud-enabled Multimedia Wireless Sensor Nodes (MWSNs), thus, for the sharing of network resources among neighboring sensor nodes in the MWSNs. It utilizes the Simplified Stochastic Path Loss (SSPL) and Shadowing (long-distance path loss mode) algorithms, to conserve the energy usage among sensor nodes in the MWSNs, while transmitting video frames to the cloud. The SSPL is deployed to determine

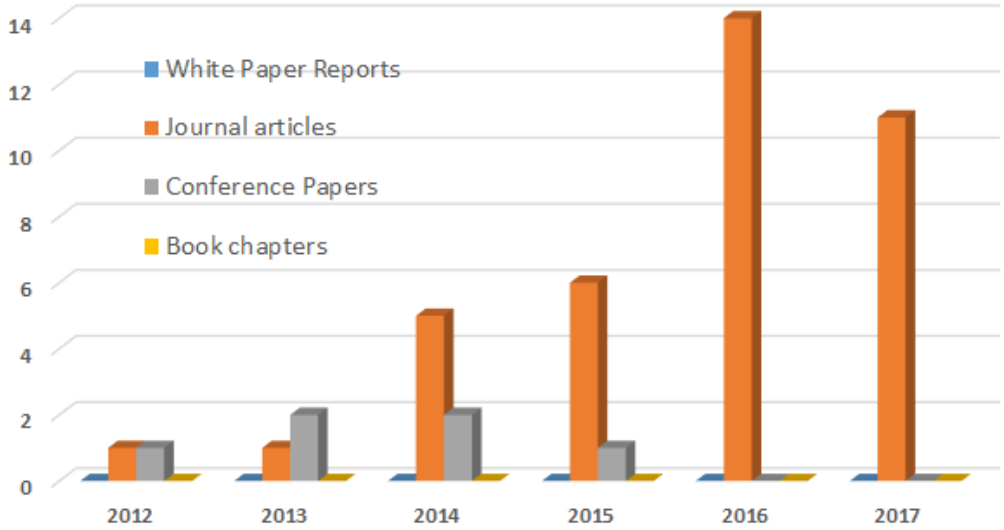


Figure 2: The distribution of the selected studies (80) by number, year and type of publication

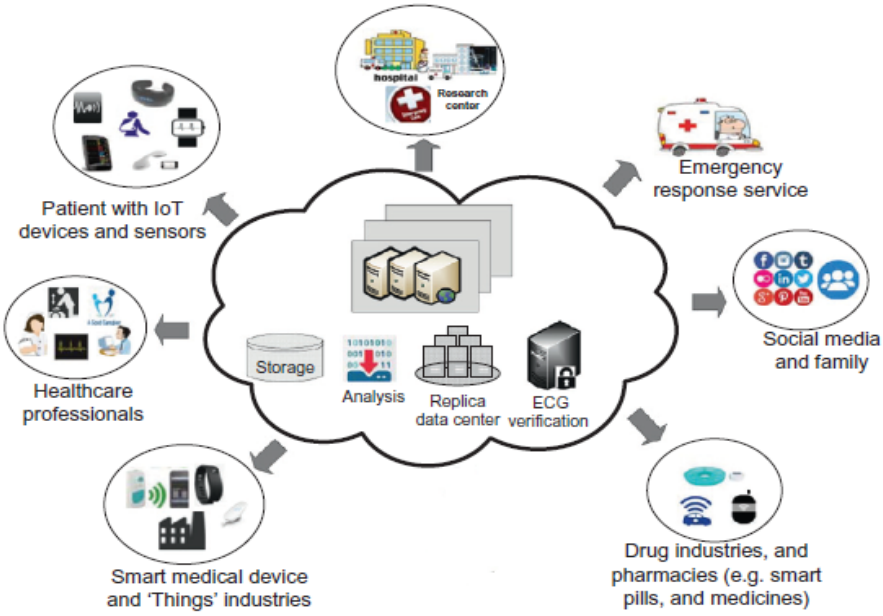


Figure 3: Health CloudIoT ecosystem conceptual illustration (Hossain and Muhammad, [29])

the suitable route path for transmission of sensed data packets from the sensor nodes to their respective sink nodes. On the other hand, the shadowing deviation algorithm is used to the suitable route path for transmission of sensed data packets from the sensor nodes to their respective sink nodes. On the other hand, the shadowing deviation algorithm is used to schedule each sensor node in the MWSNs, hence, to transmit their sensed data one at a time, in order to avoid sensory data loss and retransmission. Furthermore, energy usage of the MWSNs is drastically conserved by limiting the admission of new sensor nodes at the link layer for transmitting its sensed data to the sink node. This leads to minimum sensor nodes to be connected to the sink node for the efficient transmission of sensed data to the cloud. Lin et al. [41] implements optimization and dynamic-programming (polynomial time) algorithms, to resolve the dynamic backlight scaling optimization problem. It minimizes the black light energy consumption when a video stream is displayed on mobile phone devices via the cloud data center. The optimization algorithm is deployed to resolve the distortion and differential constraints in video streaming, while the dynamic algorithm and its polynomial-time are deployed to resolve the dynamic backlight scaling problem, by considering the minimum energy usage to display the video frames within a specific range. Consequently, the optimization algorithm runs on the cloud server which provides the dynamic backlight scaling as service to mobile phone end users.

The Dynamic Stochastic Control algorithm is proposed by [28]. Thus, managing the retrieving of the sensory data packet from medical cloud data centers to medical IoT devices (e.g. mobile phones, laptops, and medical handheld machines). IoT devices are used by healthcare professionals to download timelymedical information from the cloud storage center in order to speed up the treatment of patients with critical health conditions. The algorithm focuses on the adjustments between the compression ratio and interruptions on a sensed data packet without considering the efficiency of energy usage of data transmission and the overall network. However, Kim [33] resolved the issue by designing a polynomial-time algorithm for energy-efficient downloading of timely packet data from medical cloud data via an access point (e.g. wireless router). It minimizes the energy usage and interruptions (delays), as well as the propagation loss by utilizing the standardized measurement of 60-GHz path-loss models. This implies that 60 GHz bandwidths are dedicated to the access points, enabling uninterrupted downloading of medical sensory data packets. Furthermore, it manages buffers by computing the amount of transmission power to be allocated to each access point, while considering their buffer backlog size and stability. Also, it enables each access point to regulate its own parameters resulting in optimal energy usage, for the downloading of sensed data packets via the connected IoT access device(s).

The research in [61] implements a Cloud-based IoT Missenard index for monitoring and measuring of indoor humidity for human thermal comfort. Thermal comfort is the reaction of the human body that neither loses nor receives heat from the specific environment. It is implemented by connecting sensor nodes to a micro-controller, which is deployed to monitor and measure the level of humidity in an indoor environment (e.g. residence). The sensor nodes send an input signal to a remote computer via the micro-controller. The Missenard index algorithm in the remote computer, converts the signal retrieved from the sensor nodes into serial data. Furthermore, it relays the calibrated data back to the micro-controller, for onward transmission to the cloud platform. End users can access the calibrated data about

the level of humidity in an indoor environment, as well as the virtualized physical features of the sensor nodes, thus, to monitor and measure the Missenard index of their own indoor environment on a real-time basis. The motivation behind this project is due to the inability to measure the thermal comfort of humans in cooled or hot weather environment.

Hossain and Muhammad [29] propose a cloud-assisted industrial IoTs enabling a framework for monitoring the health of patients enabling electrocardiogram (ECG) and other related health data signals to be retrieved from sensor nodes and mobile device(s), thereafter, dispatching the sensed data signals into the cloud securely. The proposed system utilizes a watermarking technique to authenticate sensed data signals obtained from sensor nodes before they are transmitted to the cloud via a network communicating channel, thus, to avoid identity theft or clinical flaws by healthcare practitioners. Furthermore, the sensed data signals obtained from electrocardiogram (ECG) are recorded with the use of a portable ECG recording device. After which, they are forwarded to a mobile phone for the removal of redundant signals and inserting watermarking on them, for onward transmission to the cloud. Sequential features are extracted from the ECG reconstructed data signal and classified using a One-class Support Vector Machine classifier. The classified watermarked ECG data signal is dispatched to prospective healthcare professionals for decision making.

Luo and Ren [44] implement an efficient Particle Swarm Optimization combined with Simulation Annealing (PSOSAA) algorithm for medical monitoring and managing cloud-based IoT applications. WBAN is attached to the patient at his/her residence for the retrieval of sensing data, which are dispatched to a smartphone, for onward transmission to the cloud platform. The proposed algorithm is implemented on a middleware residing on the cloud resource database. The particle swarm optimization (PSO) searches for the actual sensor among multiple seamless sensor nodes that can deliver the required data requests by determining the proximity between potential sensor nodes. On the other hand, simulated annealing enhances the performance of the PSO by increasing its searching ability, thus, to obtain the desired sensor nodes for the acquisition of sensed data in real time and the continuous monitoring of patient health.

Sareen et al. [65] propose a cloud-based enable IoT framework for the detection and monitoring of Ebola outbreak at early stages, to curtail the spread of the disease. The IoT-based cloud framework has the potential to observe the present status of the outbreak and detect infected users that have the tendency of spreading the disease. It is composed of a WBANs, radio identification device, mobile phone, and the cloud data center. A patient or client is registered into the system via his/her mobile phone by entering personal and contact information. The system in turn automatically generates a unique identification number for each user. The WBANs and RFID tags are attached to the body of the registered clients. Sensed data is constantly retrieved from WBANs and stored in the cloud data center for detailed analysis. During analysis, the pool of sensed data is categorized into six sections with the aid of a J48 decision tree algorithm in accordance with the status of the symptom. A decision tree-based algorithm is used to graphically display the classification process of given EBoV attributes for a specific category. Constant monitoring using the WBAN and RFID is performed on users based on the outcome of categorization. Furthermore, the temporal network graph algorithm is adapted to indicate the proximity between infected and uninfected users. If their proximity is close between them, then the uninfected user is alerted via its mobile phone in order to avoid close contact with the infected user.

Yang et al. [80] implement a cloud-enabled IoT based on electrocardiogram (ECG) wearable sensors for monitoring patients in order to aid the diagnoses of heart diseases. ECG WBANs on patients transmit sensed data to smartphones, and in turn, is forwarded to the cloud platform via a wireless communication channel. The sensed data are refined by extracting noisy or redundant signals with the aid of a machine learning algorithm, in order to speed up the diagnoses of heart diseases.

Shi et al. [66] implement a deep learning algorithm that combines the features of deep convolutional neural networks on multichannel time series and convolutional auto-encoder, thus, for detecting and monitoring of human fatigue. Sensed data (e.g. heart rate, body temperature, oxygen, and blood sugar) retrieved from WBANs are transmitted to mobile phone device(s). The physiological sensed data is further transmitted to the cloud platform by the mobile device(s). Consequently, the sensed data are filtered and transformed into a knowledge based in the cloud, to ascertain the severity of individual fatigue in a real time basis. Individuals are alerted to keep themselves away from danger as well as prompting healthcare personnel to take a proactive medical care on victims with fatigue. Jutila [32] proposes an edge router for the interconnection of multiple IoT device networks. Regressive Admission Control (REAC) and Fuzzy Weighted Queuing (FWQ) algorithms are deployed in the edge router to monitor and determine the network quality of service changes within the multiple IoT device networks. The FQW algorithm is used for improving traffic path for the smooth transmission of sensed data from the sensor nodes to the edge router. Sensed data can be accessed by clinicians from the edge router. Consequently, the edge router transmits the sensed data to the cloud data center. Performance of end-to-end networks is managed by the REAC algorithm for preventing malicious traffic or denial of service attack during data transmission. Hence, resulting in a minimal use of bandwidth communication.

Abawajy et al. [1] personal server, and the cloud data center. Each sensor node is capable of collecting data, aggregate and perform basic processes on the data before transmitting it to a personal server. The SMO-based classification algorithm performs basic analysis and aggregation on sensed data and triggers an alarm signal, to either notify authorized patients and emergency health personnel for the availability of sensed data to be accessed in real time or dispatching the data into the cloud. Furthermore, energy consumption of the sensor nodes and the transmission bandwidth rate are minimized by the usage of fuzzy-based data fusion algorithm, which extracts the significant sensed data from the redundant data.

Kumar et al. [36] propose a multi-objective particle swarm optimization (MOPSO) algorithm in the cloud broker system

Table 1: The deployment of algorithmic techniques for CloudIoT applications in health domain

S/N	Authors Name/Year of Publication	Algorithm	Challenges Resolved	Results	Programming Lang.	Simulation Package	Network Comm. Channel	Gateways
A	Abawajy et al. [1]	i)SMO-based Classifier Technique ii) Fuzzy Data Fusion Algorithm	Real-time sensory data redundancy and depletion of sensor network lifetime	Minimize bandwidth transmission rate of sensor networks and improve battery lifetime.	C++	BIDMC Congestive Heart Failure Database (CHFD), ECG Sensor Emulator	WIFI Bluetooth	Mobile phone
B	Kim et al. [33]	i) Polynomial-time Algorithm ii) Dynamic Buffering Technique	IoT sensing device(s) energy consumption, and propagation delays leading to loss of data	Efficient uninterrupted transmission of sensory data from the access point to destination at reduced energy usage.	Not Specified	Monte Carlo Simulation	60 GHz WIFI Channel	Not Specified
C	Diallo et al. [21]	i) Real-time Query Optimization Algorithm ii) Convex Cost technique	The problem of real-time data redundancy and in-efficient transmission rate from sensor nodes to the cloud	Reduces energy and improved computational and transmission latency.	C++	OPNET (Optimized Network Engineering Tools)	WIFI	N/A
D	Mendes [46]	i) Simplified Stochastic Loss Algorithm ii) Shadowing Algorithm	Computational processing of huge multimedia sensed data	Speed up the processing of the multimedia sensory data for timely access by application request.	C++	OMNET++	WIFI	N/A
E	Lin et al. [41]	i) Optimization (Dynamic Programming) Technique	Scaling optimization problems considering the distortion, differential and time complexity constraints	Timely provisioning of dynamic backlight scaling with minimal energy consumption as services.	MATLAB	Cloud-based energy-saving service Chunghwa Telecom (CHT) cloud	60 GHz WIFI Channel	Not Specified
F	Hong and Kim [28]	i) Dynamic Stochastic Control Algorithm	To determine the compression rate ratio unit (RU) in sensor network nodes	Optimized the compression rate of each ratio unit	Not Specified	Monte Carlo Simulation	WIFI	Not Specified
G	Luo and Ren [44]	i) Hybrid Particle Swarm Optimization with Simulated Annealing Algorithm	To search for the suitable sensor nodes in WSNs considering their proximity	Improves searching ability and thus obtain actual sensor nodes to actualize the desired user requests.	Java	CloudSim	Bluetooth WIFI	Mobile Phones
H	Shi et al [66]	i) Deep Convolution Learning Algorithm	The provisioning of sensing data for detecting and monitoring of human fatigue	Improves timely access to sensed data for the prediction of human fatigue.	MATLAB	N/A	WIFI 4G Zigbee	Base Station
I	Yang et al. [80]	Machine Learning Algorithm	Extraction of noisy and redundant sensing signals	Refined useful sensed data provisioning based on end user requests.	MATLAB	N/A	GPRS Zigbee Bluetooth	Mobile Phones Base Station

Table 1: The deployment of algorithmic techniques for CloudIoT applications in the health domain (cont.)

S/N	Authors Name/Year of Publication	Algorithm	Challenges Resolved	Results	Programming Lang.	Simulation Package	Network Comm. Channel	Gateways
J	Jutila [32]	i) Regressive Admission Control (REAC) Algorithm ii) Fuzzy Weighted Queuing (FWQ) algorithm	Monitoring of network performance and how malicious network traffic can be prevented during sensory data transmission	The transmission route path is improved leading to enhanced bandwidth communication rate and efficient allocation of sensory resources.	C	Network Simulator V2 (NS2)	3G	Base Station
K	Kumrai et al [36]	i) Multi-objective Particle Swarm Optimization (MOSOP) Algorithm	The inability to search for the best configuration IoT cloud service providers at reduced pricing	Timely searched for suitable IoT Cloud service provider among its competitors at reduced pricing and minimal energy usage.	Object Oriented Java Programming Lang.	JMetal	WIFI 4G	Cloud Broker Server
L	Hossain and Muhammad [29]	i) One-class Support Vector Machine (OCSVM) Classification Algorithm ii) Watermarking Technique	The issue of feature extraction and the privacy of sensory data	Sequential features were extracted from ECG data signal and authenticated with watermarking technique to prevent identity theft and loss of sensed data.	MATLAB	Amazon Elastic Computing Cloud(EC2)	WIFI	Mobile Phone, Base Station, Remote and Database Server
M	Partha Pratim Ray [62]	i) Missenard Index Technique	The inability to monitor human thermal comfort in an Indoor environment	The humidity for human thermal comfort is effectively and efficiently measured using the sensed data retrieved.	Wiring Programming Lang.	Arduino IDE 1.0.3	WIFI	N/A
N	Sareen et al. [65]	i) J48 Decision Tree Algorithm ii) Temporal Graph Technique	The inability to detect Ebola disease in real time basis and determining the distance between unaffected and affected persons	Improves the detection of Ebola by classifying related symptoms and monitoring the proximity between affected and unaffected persons.	MATLAB	WEKA (3.6) and Gephi 0.9.1 (Graph TNA)	Bluetooth WIFI 3G/4G GPRS	Mobile Phone

for IoTs distributed network to maximize the profit of the cloudbroker and minimizing the response time between clients and multiple cloud providers. The proposed algorithm is implemented on the cloud broker to perform its optimization process of searching the best configuration between the clients and the cloud providers. In other words, it assists the cloud broker to determine the best possible cloud provider among multiple cloud providers that can execute client requests in the shortest timeframe resulting in the optimal energy consumption of the IoT device(s), while maximizing profits.

2.2 Manufacturing Domain

The deployment of the CloudIoT system in the manufacturing domain is gaining more stride over the last six years as production equipment/machinery, raw materials and personnel are attached with RFID tags and sensor nodes for speeding up production processes, for the monitoring of finished products during the distribution chain to wholesalers and retailers. The production equipment is monitored to predict the status of their condition in order to prevent unforeseen breakdown during the processing stage. With the aid of IoT sensing device(s), information regarding the performance level of each production personnel or worker can be obtained in a real time basis for the purpose of staff performance evaluation and the entire growth of the establishment. Figure 4 below shows the architecture of manufacturing a CloudIoT system.

Wang et al. [77] introduce the NTGO Algorithm that enables multiples of mobile sensing devices attached to produce components for the transmitting of sensed data to the cloud. It allows all the mobile sensing devices to compete for a communication channel simultaneously, which limits the data transmission rate to the cloud due to traffic congestion. It is also challenged with computational complexity and superfluous overhead that leads to potential inaccurate decisions. Therefore, Kim [34] implements a nested game

algorithm to resolve the aforementioned issues. It uses the bargaining power and low pricing technique to actualize efficient offloading of sensory data from mobile sensing devices to the cloud platform. Each mobile sensing device(s) has the potential to decide whether to perform computation on the sensed data retrieved or dispatch it directly to the cloud. If the decision is to offload the data to the cloud, the mobile sensing device has to bargain with its counterparts that intend to offload their data to the cloud as well. At this point, the one with the highest bargaining power is allowed to offload its data to the cloud first before others. The test result shows that the nested game algorithm is practically better than the NTGO for offloading and performing computation at limited energy consumption, also, at reduced execution time and limited data packets lost during transmission between the mobile sensing device(s) and the cloud.

Zhu et al. [85] developed a TPSS scheme for the integration of IoT sensing devices and mobile cloud. The scheme consists of Time & Priority based Selective Data Transmission (TPSDT) and Priority-based Sleeping Scheduling (PSS) algorithms to obtain useful sensed data from desired sensor nodes based on mobile phone end user requests via the cloud. It enhances the reliability of the sensor nodes by conserving their energy usage, while satisfying mobile end users data requests. The TPSDT algorithm is responsible for the selection of suitable sensor nodes for the retrieval of the desired sensed data, in accordance with the time and priority features of the data requested by mobile users. The features of data requested by mobile users are recorded in a Point Time Priority (PTP) table that resides in the cloud data center. Thus, the data requests with high priorities are forwarded first, followed by those with lower priorities to the cloud. On the other hand, the PSS algorithm initially integrates the time and priority features of the requested data from mobile end-users into the sensor nodes sleep scheduling process. Hence, to obtain and transmit meaningful data to the cloud with the PTP table via its corresponding gateway. It awakes only the set of sensor nodes that have more battery energy for the retrieval of sensed data in each time period, therefore, putting other sensor nodes to sleep, in order to conserve energy consumption.

Qu et al. [59] propose a cloud manufacturing enabled IoT real-time production logistics synchronization system. By automating both external and internal production resources to facilitate production planning processes leading to effective and efficient finished products. Internal production resources such as forklift, scheduling/processing machines, transportation vehicles, and raw materials are fitted with sensor nodes and RFID tags. External resources, such as raw materials ordered and registered suppliers delivery vehicles are also attached with RFID tags. Data collected from both internal and external production resources (via sensor nodes and RFID tags) during and after the production phase are continuously transmitted to gateways. The gateways consequently dispatch the data on a regular base into the cloud. Users can access the cloud with their mobile phones and PCs via the internet to place an order for a product or send customized requirements. If the product ordered for is available, it will be delivered to the client immediately as long as payment and other service level

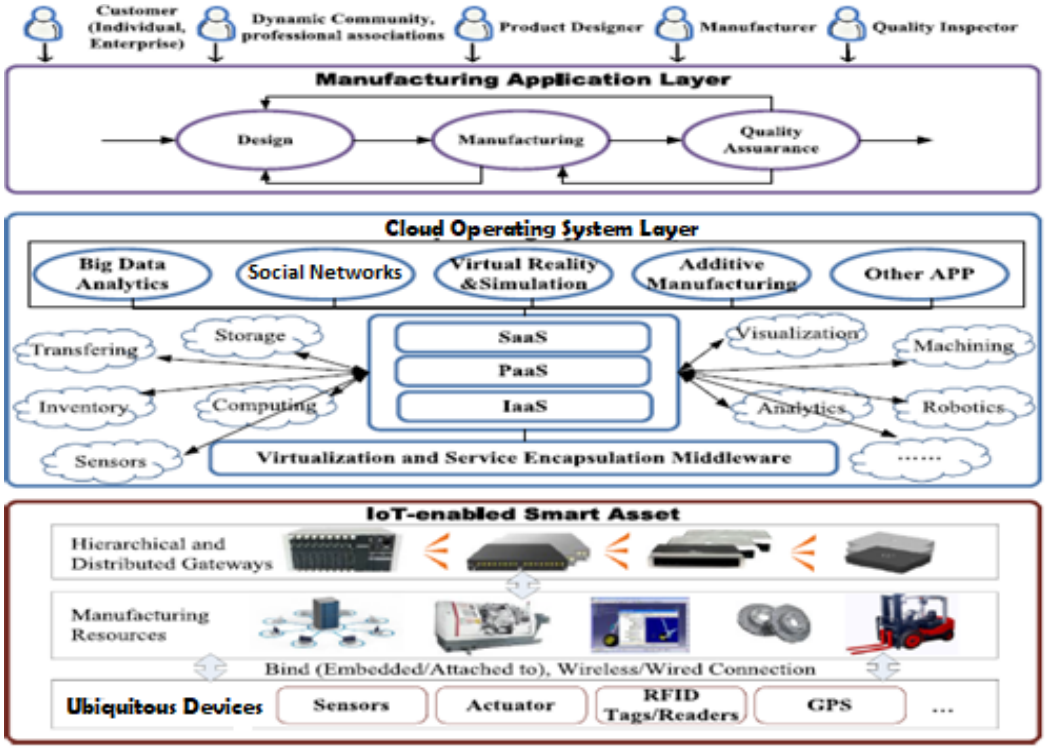


Figure 4: Manufacturing CloudIoT system architecture (Yang et al., [77])

agreements are met by the client. However, if the product is not available, the cloud automatically checks for available resources (both internal and external production) from its pool of sensed data and triggers the production process of the client's requests. Peradventure they run out of raw materials in stock to be used for the production of the pending request, an order is automatically forwarded to a registered supplier. The order is monitored until it is delivered in the production warehouse.

An IoT enabled dynamic service selection framework across multiple manufacturing cloud providers is proposed by [80] in order to monitor the status of services employed to execute tasks on a real-time basis for the prediction of task progress. It consists of a manufacturing broker cloud that interconnects multiple services from manufacturing cloud providers. The broker cloud monitors the status of available services and the volume of user task requests in advance. Enabling the optimal selection of services from multiple manufacturing cloud providers. Changes from both the market (Manufacturing cloud providers) and consumers are also captured using the event-driven dynamic service selection, which triggers the optimization of service selection cloud providers. For instance, the breakdown of a machine may occur during the execution process leading to delay of task completion. On the other hand, if there are changes in user task request, for example, the user request is altered to accommodate additional requirements, which will need additional services for it to be executed. Thus, these changes can be predicted in advance with the aid of sensor nodes that are connected to the MC broker leading to an optimal selection of services that will effectively and efficiently execute the given user sensing task requests. Georgakopoulos et al. [25] propose a programmable scheme at the cloud manufacturing network edge such as smart gateways (mobile phones) that are closer to the source (IoT sensing devices) of data collection for the provision of an efficient real-time prediction and key performance indication (KPI) monitoring. The cloud manufacturing network edge is based on Raspberry Pi 3, UDOO board and ESP8266 gateways, to store and process sensory data from production power plants on a smaller scale in real time basis.

Yang et al. [79] implement a cloud-based enabled IoT assisted manufacturing of product customization and personalization. It is composed of social networking platforms, IoTs sensing devices (RFID tags and sensor nodes) and the cloud data center. The social networking platforms are deployed for the collaboration between multiple individuals or group of skilled personnel to aid open innovations. RFID tags are attached to manufacturing parts, materials and other related production facilities for real-time monitoring during production phases. The cloud provides virtual physical manufacturing parts and materials needed for the production of the customized product in accordance with the specification received. It also obtains virtual features and status of production parts and materials from RFID readers whose tags are attached to the physical components of manufacturing facilities and raw materials.

Narman et al. [48] propose the scheduling of IoT applications in the cloud platform. It utilizes the scheduling algorithm to schedule servers that reside in the cloud data center. The servers are deployed for the processing and storing of heterogeneous and homogenous IoT sensory data, thus, to improve the performance metrics of the CloudIoT platform by considering the drop rate, throughput, and the IoTs sensory device(s) energy consumption rate. Furthermore, priority algorithm is adapted based on a first-come-first serve technique, which uses the parameters such as arrival rates of IoT requests and service rates of servers, hence, to determine which request needs to be executed first before the other requests. It considers the shared and dedicated servers for updating service rates dynamically for each IoT applications and sensed data requests.

2.3 Agricultural Domain

CloudIoT is deployed in the agricultural sector for two basic reasons namely to predict the growth of plants for optimal crop yield through constant monitoring and to control the farm irrigation system for the efficient supply of water to the soil-plant as required in real time basis. Roopaei, et al. [64] propose a CloudIoT infrastructure to monitor irrigation scheduling and water resources on soil using thermal sensing nodes. This research is motivated due to the poor irrigation scheduling and inefficient utilization of water resources resulting in poor farm produce. The thermal sensing nodes have the capability to sense emitted temperature signals from the plant and not from the soil, making it preferable and more efficient than other existing schemes. Thermal sensing is utilized to monitor water stress in order to limit the level of water stress by displaying its temperature level. Data from the thermal imaging sensor nodes are transmitted to the cloud data center through a high-speed communication channel such as the WIFI and LTE for energy optimization. Furthermore, the data is processed in the cloud ready for delivery to clients (farmers). With this information in hand, farmers can effectively and efficiently schedule irrigated water to the soil, in order to achieve high crop yield. See Figure 5 for more insight.

2.4 Smart Environment Domain

The idea of making the human environment to be more conducive and habitable is due to the increase in population growth and crime rate in some major cities across the globe. Academic researchers and environmentalists are currently embracing the utilization of CloudIoT system as a solution to make the populated environment smarter and conducive for habitation. There are various CloudIoT application systems deployed in some major cities which include car parking systems, surveillance systems, CloudIoT based disaster monitoring systems, smart homes and building systems, just to mention a few. A detailed analysis of previous research in the deployment of CloudIoT infrastructure is presented as follows.

A Cloud-based sensor architecture for controlling and monitoring the physical environment is introduced by [19]. It utilizes the context-oriented approach to obtain sensory data from sensor nodes, without considering their physical features and energy consumption. The context-oriented approach

adopted the Open Service Gateway Initiative (OSGi), for dispatching sensory data from Wide Area Network (WAN) to Local Area Network (LAN). It also creates the enabling platform for service providers, developers and software vendors to implement, deploy and manage sensory services. However, [23] introduce the Cloud4Sens architecture that utilizes the data-centric and device-centric models to control and monitor the events of sensor nodes that are embedded in the environment. The data-centric model retrieves sensed data from sensor nodes, embedded in a specific geographical region at a given time interval. There are two types of manager agents in the cloud4sense namely the Data Manager Agent (DMA) and the Infrastructure Manager Agent (IMA). The data manager agent is responsible for

Table 2: The deployment of algorithmic techniques for CloudIoT applications in manufacturing domain

S/N	Authors Name/ Year of Publica-tion	Algorithm	Challenges Resolved	Results	Programming Lang.	Simulation Package	Network Comm. Channel	Gateways
A	Narman et al. [48]	i) Server Scheduling Technique ii) Priority-based Algorithm (FIFO)	Conventional cloud unable to provide storage services for the huge volume of sensory data	Efficient provisioning of storage services to sensed data by improving drop rate, computational latency, and throughput.	MATLAB	CloudSim	Ethernet	Server Scheduler
B	Wang et al. [77]	i) Nested Two-stage Game-based Optimization (NTGO) Algorithm	Provisioning of sensing resource at the expense of high rate of the bandwidth transmission rate	The provision and offloading of reliable sensing request data into the cloud.	OTEL	NS^2	Not Specified	N/A
C	Kim [34]	i) Nested Game-based on Computation Offloading Technique	The inability to offload sensed data which increases the overhead signaling and selection of a mobile phone sensing resource to satisfy numerous sensing task requests	Efficient quality of service delivery of sensed data to end users at reduced pricing and sensor nodes energy usage.	C	Network Simulator V^2 (NS^2)	4G WIFI	N/A
D	Zhu et al. [85]	i) Time & Priority based Selective Data Transmission (TPSDTP) ii) Priority-based Sleep Scheduling (PSS)	Inefficient provisioning of sensed data to vital end user requests and high energy consumption of sensing device(s)	Improves the efficiency of sensed data provisioning by satisfying important request first before others while conserving the energy usage of IoT sensing device(s.).	MATLAB	Based on Assumption	WIFI	Not Specified

Table 2: The deployment of algorithmic techniques for CloudIoT applications in manufacturing domain (Cont.)

S/N	Authors Name/ Year of Publica-tion	Algorithm	Challenges Resolved	Results	Programming Lang.	Simulation Package	Network Comm. Channel	Gateways
E	QU et al. [59]	a) Smart Object Layer b) Smart Gateway Layer c) Service Layer d) Application Layer	N/a	N/a	Not Specified	Not Specified	WIFI 4G GPRS Bluetooth	Mobile Phones
F	Yang et al [80]	i) Event-driven Dynamic Service Selection	The Inability to determine the best cloud service provider (CSP) among CSPs	Optimal selection of the desired CSP among manufacturing cloud service providers to satisfy the user request.	N/A	N/A	WIFI Bluetooth	Broker Cloud
G	Yang et al [79]	i) Dynamic AID Allocation Algorithm ii) Compressed Bitmap Algorithm	Uplink and downlink traffic metering control of sensed data	Reducing communication overhead and prolong the lifetime of IoT sensing device(s).	N/A	N/A	WIFI	N/A
H	Georgakopoulos et al [25]	Real-Time Prediction Key Performance Indicator (KPI) Monitoring Algorithm	The issue of increase in bandwidth rate used for the transmission of sensed data from IoT sensing devices to the cloud and high energy consumption of IoT sensing devices	The use of Raspberry Pi closed to sensing devices leads to the reduction of bandwidth rate usage and improved transmission speed while conserving energy usage of IoT sensing devices.	N/A	N/A	Bluetooth WIFI	Raspberry Pi3 UDOO ESP8266

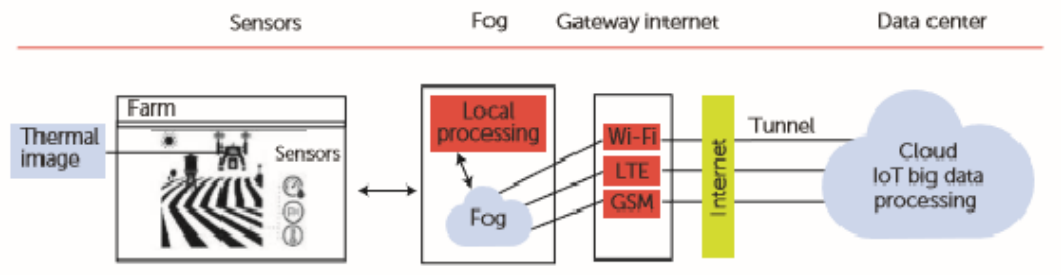


Figure 5: The overview of cloud of things system in irrigation farming [48]

Table 3: The deployment of algorithmic techniques for CloudIoT applications in the agricultural domain

S/N	Authors Name/Year of Publication	Algorithm	Challenges Resolved	Results	Programming Lang.	Simulation Package	Network Comm. Channel	Gateways
A	Poopaei et al. [64]	N/A	Poor irrigation scheduling and inefficient utilization of water resources on farm crops	Improves the scheduling of irrigation and utilization of water in real time basis.	N/A	N/A	WIFI 4G LTE	Raspberry Pi

coordinating end user subscriptions to obtain a particular type of sampling event or sensed data over a specific geographical area. On the other hand, the IMA manages the formulation of the virtualized sensing features, by submitting a set of task requests to the physical sensor nodes. In addition, the cloud4sens provides the sensed data retrieved via the data-centric model as PaaS, and that from the device centric model as IaaS to end users and cloud service providers.

Mitton et al. [47] implement an architecture that enables internet users to have access to data retrieved from heterogeneous and dynamic sensor nodes via the cloud platform. The architecture consists of three main layers namely Hypervisor Block, Autonomic Enforcer, and Volunteer Cloud. The hypervisor block is assigned with the task of abstracting sensing data from sensor nodes. On the hand, the autonomic enforcer is a channel that connects the virtualized sensor nodes to the cloud, thereby exposing their functionalities as services to end-users via the internet. Consequently, the volunteer cloud manager enforces the combination of dynamic functionalities (resources) of virtualized sensor nodes in the cloud platform. Data obtained from the sensor nodes are formally transmitted to a local relational database server, thus, enabling end users that are close to the local database server to retrieve desired information. Furthermore, the relational local database server is decolonized and distributed, in order to store and rapidly dispatch sensed data in a transparent manner into the public cloud, enabling end users across the globe to have access to the information irrespective of their geographical locations. Zhu et al. [84] implement a collaborative location based sleep scheduling algorithms for IoT sensing device(s) enabled mobile cloud platform. The algorithms determine the awake and sleep state of the sensor nodes in order to reduce their energy consumption. Mobile phone end users send their request to the cloud via the internet. Upon receiving user requests, the CLLSS1 algorithm tracks the location of the mobile user with the aid of Global Positioning System (GPS) and forwards the user location information to the StarTrack server that resides in the cloud. With this information in hand, the CLSS2 algorithm awakes the embedded desired sensor nodes that can fulfill the required user requests, while putting other neighboring sensor nodes to sleep. Furthermore, the awake sensor node automatically goes to sleep after fulfilling the required sensed data request. Theoretical and simulation results show that both algorithms enhance the lifetime of sensor nodes as well as satisfying mobile end users requests efficiently.

Paul et al. [53] presented a conventional Alternating Direction Method of Multipliers (ADMMs) model, based on distributed consensus-based estimation algorithm for CloudIoT. It performs limited iterations and minimized communication overhead per iteration between the cloud and the IoT sensing devices while compromising the estimation error. However, Abdelwahab et al. [2] introduce a virtualization scheme to manage virtual sensor nodes from selected gateways for the efficient execution of an on-demand task request with further reduction of overhead communication while avoiding estimation error. The virtualization scheme is composed of

three algorithms namely Gossip-based, Randomized & Asynchronous Distributed Virtualization (RADV) and Randomized & Asynchronous Distributed Estimation (RADE). The gossip-based algorithm enables the discovery of sensing data by using a gossip policy for transmitting sensing request tasks from the cloud to the sensor gateways, also, enabling the selection of the desired sensor nodes to execute the tasks. RADV algorithm is responsible for the virtualization of the selected sensor nodes for the execution of the sensing tasks. On the other hand, RADE algorithm is utilized to estimate a set of unknown parameters among asynchronous IoT sensing devices without considering their harmonization. Simulation results show that the virtualization scheme reduces communication overhead significantly without compromising the estimation error, unlike the ADMMs. This motivated Ali et al. [5] to develop a scalable and lightweight intelligent distributed surveillance system that makes use of the integrated framework of IoT and cloud computing. The system is composed of cloudlets sited close to the to the Wireless Camera sensor nodes (WCSNs), to reduce communication distance to the cloud for optimal bandwidth transmission rate. Furthermore, Alsmirat et al. [6] introduces a mobile edge (MEC) computing gateway between the cloud and the WCSNs for optimal bandwidth transmission rate and to curtail the entire video surveillance distortion. A Simplified dynamic effective Airtime estimation (SDEAE) algorithm is implemented in the MEC to determine the sum ratio of data dropping to the physical rate of each camera sensor nodes while retrieving streamed video data for onward transmission to the cloud. Streamed video data can be deleted from the MECs, upon confirmation of reception by the cloud server. The bandwidth transmission rate is allocated to each MEC server that is connected to the cloud and the number WCSNs that are connected to a specific MEC server. This is actualized by estimating the total amount of streamed video data received from each MEC server, which signifies the total number of WCSNs attached to each corresponding MEC server. Therefore, the more streamed video data is received from a specific MEC server by the cloud data center, more bandwidth is then allocated to that MEC from the cloud bandwidth.

Madria et al. [45] implement a cloud of virtual sensor nodes for the monitoring of a specific geographical area. It comprises three layers which include client, sensor-centric and middleware-centric. The client-centric layer enables authorized users to access sensed data and the virtual physical features of sensor nodes. On the other hand, the sensor-centric layer is made of the sensor nodes embedded in a wide geographical area that is connected to the cloud data center via a wireless network communication channel. The middleware deals with service negotiation, maintaining and providing virtual sensing events and transmitting data from the sensor-centric to the client-centric layers, while considering the energy usage of the sensor nodes. It is also responsible to match the client request to the data obtained from sensors and enable multiple user requests to be satisfied by a single sensor node simultaneously, thus, minimizing the energy consumption of the sensor nodes. Lawson and Ramaswamy [38] implement a data quality and energy management tradeoffs in Sensor Service Cloud enabling consumers to access the desired quality of sensed data while reducing the energy consumption of the sensor nodes. The proposed system is composed of a network controller (Gateway). The energy optimizer is used to regulate the frequency levels either up or down depending on the status of the sensor node(s). On the other hand, the network controller houses the features of each sensor nodes and selects the best matching feeds. This is actualized by the Energy-efficient Throttling algorithm, which determines the optimal sensor nodes that can satisfy consumer requests. The network controller also utilizes the adjustable motes with the aid of a dynamic scaling algorithm to adjust the settings of the throttled.

Pham et al. [55] propose a Cloud-enabled Smart-Parking System based on the IoT-Cloud. It adopts a queuing algorithm to search for car parks with free parking slots at a minimum cost, referring car drivers to another available car park that has free parking slots when the current searched car park is fully

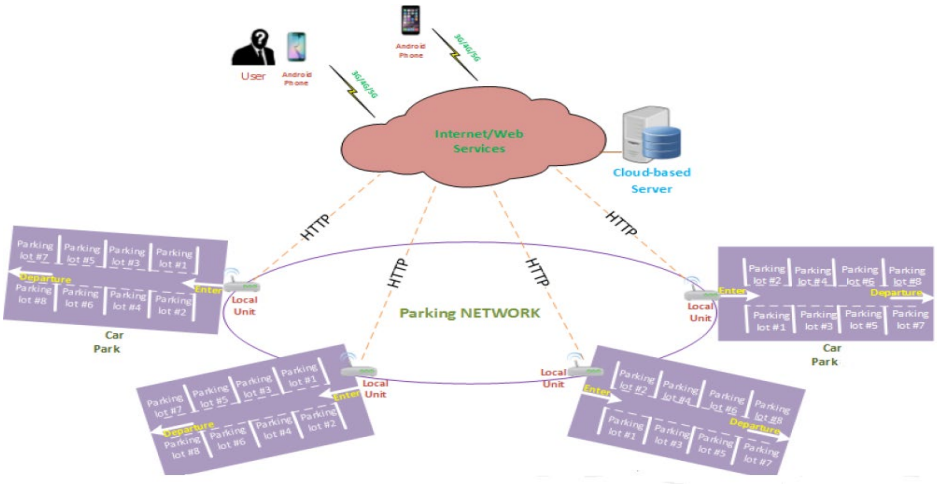


Figure 6: The Overview of a CloudIoT based smart parking system Pham, et al. [55]

occupied. Consequently, it updates the system automatically in real time to notify the current status of each car park. The parking spaces are fitted with RFID tags for monitoring and RFID reader based on the Arduino module to collect information about the current status of parking spaces for onward transmission to the cloud data center via an internet connection. Users access the system using their smartphone configured and installed with a software running on the Android operating system as depicted in Figure 6 in order to search for an available car park that has free parking space and make a reservation in advance.

Liu, et al. [42] propose a Green Data Centre with IoT sensing and cloud-assisted smart temperature control system. It is composed of Temperature Control System and a Cloud Management Platform. The temperature control system is made up of numerous sensor nodes attached to cooling systems such as the air conditioning, ventilation and outside the cloud data center room. Sensor nodes deployed inside and outside the datacenter room monitors the level of temperature and humidity. Sensor nodes installed on the cooling systems monitor the operating conditions for possible device weakness or failure. Furthermore, the entire sensor nodes transmit their sensed data signals to a gateway via the communication network channel for onward transmission to the cloud. A multilevel intelligent temperature control algorithm residing in the cloud detects the temperature level both inside and outside the data center room. If the outside temperature is detected to be relatively lower than inside the data center room, the ventilation system is turned on automatically to let in natural cold air, while the cooling systems inside the data center room are turned off automatically. On the other hand, if the temperature inside the data center room is observed to be higher than the outside temperature, ventilation is turned off compelling the cooling systems inside the data center room be turned on automatically.

Atif et al. [7] propose a cloud-enabled IoT sensing system to control and predict the availability of parking spaces in various street blocks. Embedded sensor nodes are deployed on the street block to monitor the availability of parking spaces, and when cars enter or exit the parking premise which minimizes the possibility of sending motorists across the same routes leading to traffic congestion while navigating towards the targeted parking space. Sensed data retrieved are housed by a middleware, which also masks the heterogeneity and distribution of the physical features of the sensors of the sensor nodes into virtualized form. It also delivers an open mediating interface to the application layer in the cloud platform. Consequently, sensory data retrieved are stored in the cloud database center and aggregated into knowledge-based information. Authorized car owners have access to the knowledge-based information from the cloud in a real time basis, to either search or book for an available parking space. It also enables searching for the safest route to enter or exit the parking premise in order to reduce traffic congestion. Dinh and Kim [35] implement an efficient interactive model for on-demand Sensing-As-A-Services of Sensor-Cloud. It enables sensed data to be transmitted efficiently to the cloud platform, therefore providing them as a service to multiple end-user applications. It uses the application request aggregation algorithm to aggregate sensing request intervals from various applications and determines a sensing interval as well as updating sensing requests. Furthermore, it selects an optimal consolidated sensing interval for a group of sensor nodes to minimize the rate of sensing events which reduces the amount of data packet transmissions from the sensor nodes, while satisfying all application sensing request interval requirements. In addition, a sensing update request-based on Adaptive Low Power Listing (composed of active adaptive and lazy adaptive mode) algorithm, is deployed to minimize the energy consumption of the sensor nodes, while satisfying sensing requests of various applications.

Yu et al. [81] propose a cloud-based enabled IoT sensing application for monitoring and managing energy forecasting of a building in real time basis. The proposed system comprises of various sensor nodes attached to all sections of the building as depicted in Figure 7 below. The sensor nodes sense the building to determine its humidity and temperature level, discomfort index, solar irradiation index, the degree of cloudiness and wind velocity the sensed data retrieved are forwarded to a local agent server with the aid of the best first algorithm, to optimize bandwidth transmission rate and conserve the energy usage of the sensor nodes. Thereafter, the sensed data are filtered and transformed into meaningful information by utilizing a machine learning algorithm in the cloud data center.

Barceló, et al. [8] propose an IoT-cloud optimization in the next generation smart environment. A mathematical efficient linear programming algorithm is adapted to resolve the issue of service distribution problem (SDP) in the IoTs-cloud system resulting in a significant reduction of overall power consumption usage sensor nodes. The formulated algorithm is based on a network flow linear programming that optimizes the dissemination of cloud services. It characterizes the sensed data retrieved via a directed fixed graph in order to generate the final improved vital information to be conveyed to end users. The fixed graph acknowledges optimal polynomial time solutions, enabling the capturing of unicast and multicast transmission.

Li et al. [39] propose a monitoring system via face recognition based on cloud IoT platform using Gabor and CS-LBP features. It is composed of a monitoring client module for detecting and recognizing faces and a cloud base storage technique for data virtualization. The monitor client (Video Camera Sensor Nodes) detect and capture face image for the recognition process. A modified AdaBoost method based on a Gabor feature algorithm implemented in Cament, et al. [13], is used for facial detection. On the other hand, the Center-symmetric Local Gabor Binary Pattern (CS-LBP) feature extraction algorithm is used to extract facial image captured for recognition from the monitor client (Video Camera Sensor Nodes). The data retrieved which includes the detected face number, location, personal identification, and other related information are dispatched to a client-server, thus, to limit the communication distance between the monitor client and the cloud the client-server transmits the data or information via the internet to the cloud for storage and further computational process. Furthermore, the data are condensed in a global map (e.g. Open

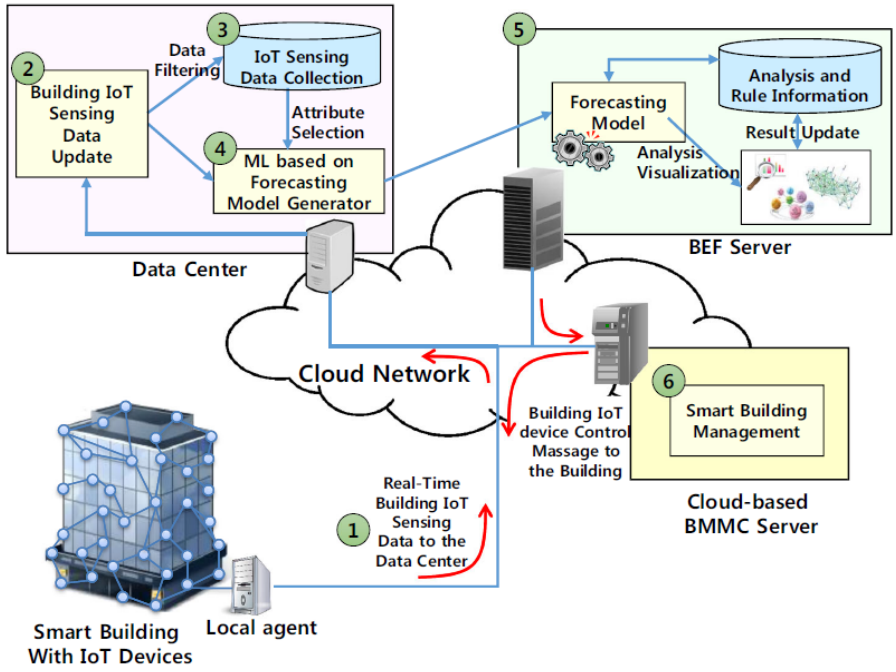


Figure 7: The model of CloudIoT based building monitoring system Yu, et al. [81]

Street Map (OSP)) for real-time monitoring in the cloud. The experimental result shows that the LBP and Gabor algorithms accurately extract orientation and structural information of the image captured by avoiding redundant sensed data.

Chatterjee and Misra [18], introduces an external and internal caching algorithm to guarantee energy efficiency in the utilization of physical sensor nodes by resolving the issue of redundant sensed data during transmission. The external cache determines the ideal caching interval that will reduce the energy consumption of the IoT sensing devices. The internal cache makes the decision to transmit request data received from user-applications, directly to the cache or re-caches the executed request data from the external cache and transmit it. Dinh, et al. [22] further optimize energy consumption of the sensor nodes by proposing a location-based interactive model. It obtains sensory data from multiple distributed sensors only on-demand, based on mobile users' location of interest. The proposed model was actualized by adapting an on-demand scheduling algorithm to enable on-demand interaction between cloud and IoT device(s). Sensing information or data is retrieved from the sensor nodes that are closely located to the mobile user requests. Sensors are placed in the inactive mode when there is incoming user demand. A clustering algorithm based on network communal discovery theory is used to fuse multiple similar sensed data retrieved from the sensor nodes.

Wang et al. [76] propose a Multimedia Sensing as a Service cloud platform for exploring the resource saving potentials at Cloud-Edge IoT, as depicted in Figure 8 below. A Truncation-Point-Optimized Resource Allocation (TORA) and QuadTree Decomposition (QTD) algorithms are adapted for managing IoT resources in the cloud-edge enabled multimedia sensor nodes to improve the prioritization-based quality experience for wireless multimedia data communications with a significant reduction of energy consumption of the sensor nodes. The proposed system uses the quadtree decomposition algorithm for the processing of a single image or inter-frame pictures simultaneously in order to efficiently prioritize these images by utilizing comprehensive premium regular concepts. The large region of the images with the number of pixels are compressed into a small number of megabytes, as well as presenting the high-frequency edges with limited pixels as a small number of megabytes. This, leads to the reduction of energy usages due to the inequality features of multimedia data provided at the edge. Consequently, multimedia data quality is optimized using the TORA algorithm to regulate the communication resource parameters (such as power and modulation) between crucial premium information and unimportant information.

Qin et al. [58] propose an architectural design of cloud-enabled IoT geological disaster monitoring, warning and post-disaster rescue system as depicted in Figure 9 to be deployed in geographical areas where the incidence of natural and geological disasters are prevalent. The proposed system tends to obtain, transmit, aggregate and process data for analysis on a real-time basis to notify in advance any possible disaster occurrence in a specific region. Also, this provides a reliable information that will aid rescue missions after disaster incidences in order to minimize human casualty and loss of property. Sensor nodes such as pluviometer, water gauge, displacement meter and crack meters are attached to strategic places such as rivers, mountains and road paths to obtain sensed data which are transmitted to a local unit-based server via 4G and GPRS communication network channel. At the local unit

server station, the GIS model efficiently processes the multiple spatiotemporal geographical sensed data retrieved from sensor nodes in order to minimize sensed data redundancy before they are dispatched to the cloud for further processing and analysis.

6 CLOUDIOT Simulation Environment

Simulators provide a useful alternative to evaluate algorithms and protocols in a controlled environment. The discrete-event simulator is mostly adopted in ClouIoT infrastructure to evaluate and predict the performance of an algorithm. In the discrete-event system, events or process of interest change value or state at a distinct point in time. According to, discrete-event simulation is made up of a collection of techniques that are utilized to study a discrete-event dynamical system, which leads to the generation of sequences called paths. These techniques involve modeling concepts that are used to abstract the essential features of the model or system to form a coherent set of precedence and mathematical relationships among its elements. The design of unique software converts these relationships into a computer-executable code that can generate the essential sample-path data and converts these data to estimate the performance of the models or systems, consequently, evaluating these estimates to determine their accuracy.

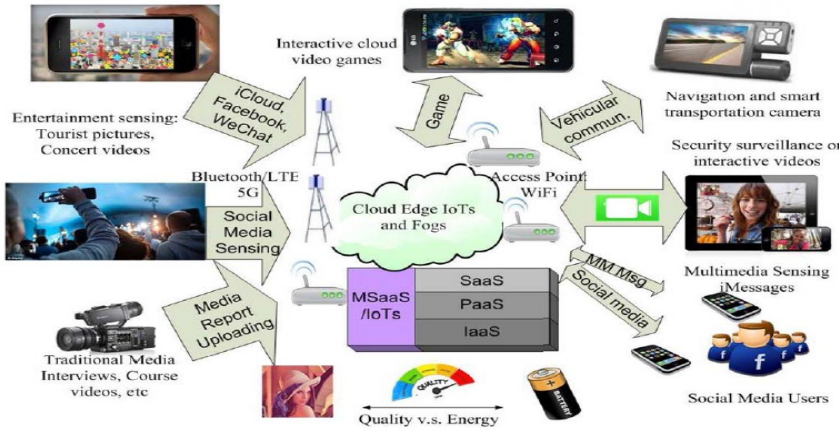


Figure 8: An overview of the MSaaS cloud-edge enabled IoT fog system (Wang, et al. [76])

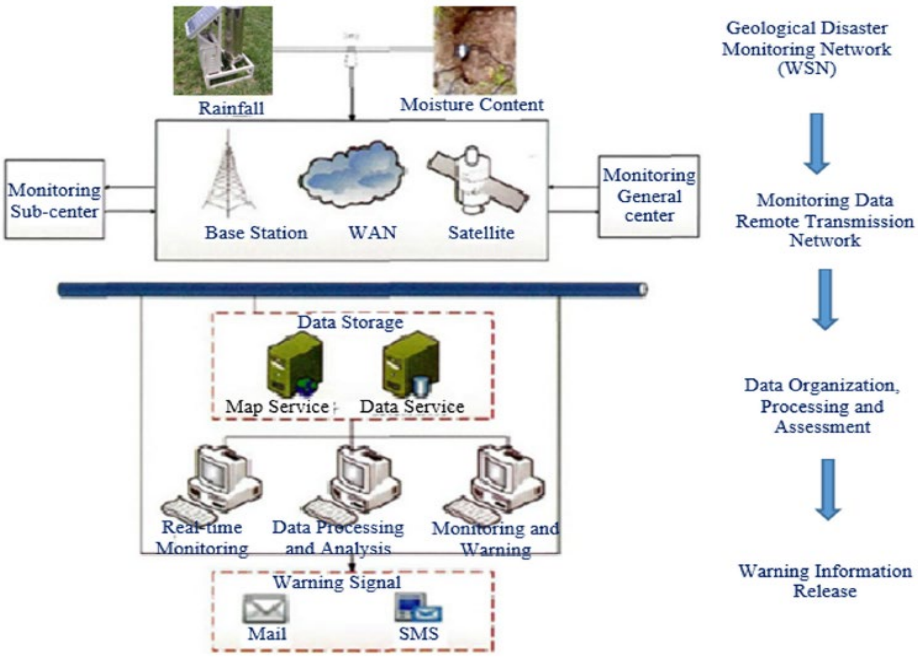


Figure 9: An architecture of CloudIoT based post-disaster warning-rescue system (Qin, et al. [58])

Table 4: The deployment of algorithmic techniques for CloudIoT applications in environmental domain

S/N	Authors Name/Year of Publication	Algorithm	Challenges Resolved	Results	Programming Lang.	Simulation Package	Network Comm. Channel	Gateways
A	Chen and Chen [19]	i) Open Service Gateway Initiative (OSGi)	Unreliable delivery of sensing data from sensor nodes to the cloud	Improves the delivery of sensed data to the cloud and provide the environment for software developers to implement IoT applications on the cloud platform as service (PaS).	N/A	N/A	WIFI	Router
B	Fazio and Puliafito [23]	i) Infrastructure Manager Agent (IMF) ii) Data Manager Agent (DMA)	The issue of poor IoT sensing device(s) management and unreliable sensed data delivery	Effective management of IoT sensing devices by conserving their energy usage and reliable sensed data delivery to the cloud.	N/A	N/A	WIFI	N/A
C	Miltton et al. [47]	i) Clustering Algorithm	The high rate of energy consumption by IoT sensing devices	Minimizes energy consumption and improves the lifetime of the sensor nodes.	ContikiOS	Contiki	XMPP	Localized Database Server
D	Zhu et al. [84]	i) Location-based Sleep Scheduling Algorithms (CLSS1 and CLSS2)	The complexity of locating the mobile phone users and inappropriate scheduling of sensor nodes	Accurate detection of mobile end users for efficient provisioning of sensory data requests and reduction of energy consumption by the sensor nodes.	MATLAB	NetTopo[2] Simulator	WIFI Bluetooth	N/A
E	Paul et al. [53]	i) Alternative Direction Method of Multiplier (ADMM) Technique	The communication overhead between sensor nodes and the least square estimation problem	Performs minimum communication overhead per iteration while sustaining the estimation accuracy.	MATLAB	Monte-Carlo	N/A	Not Specified
F	Abdelwahab et al. [2]	i) Gossip-based ii) Randomized & Asynchronous Distributed Virtualization (RADV) iii) Randomized & Asynchronous Distributed Estimation (RADE)	The determination of reliable sensor nodes for specific task request execution with increased communication overhead	Efficient selection of suitable sensor nodes for task execution while reducing the communication overhead, therefore guaranteeing the sensor nodes lifetime.	MATLAB	Monte-Carlo	WIFI 4G	Mobile Phones, Computer Server

Table 4: The deployment of algorithmic techniques for CloudIoT applications in environmental domain (cont.)

S/N	Authors Name/Year of Publication	Algorithm	Challenges Resolved	Results	Programming Lang.	Simulation Package	Network Comm. Channel	Gateways
G	Ali et al. [5]	i) Intelligent Distributed Technique	The limitation bandwidth transmission rate between sensing devices and the cloud	Improves transmission rate by adopting cloudlets which serves as temporary storage close to the sensing devices.	N/A	N/A	WIFI 4G	Cloudlet
H	Alsmirat et al. [6]	a) Simplified Dynamic Effective Airtime Estimation (SDEAE) Algorithm	The limitation of bandwidth transmission rate between sensor camera devices and the cloud.	The optimal bandwidth transmission rate was realized leading to a reduction of video surveillance distortion.	C++	Network Simulator V^3 (NS^3)	WIFI 4G	Mobile Edge Server
I	Madria et al. [45]	i) Client Centric, Middleware, and Sensor-Centric	The issue of untimely delivery of client sensory data requests leading to the high energy consumption of the sensing devices.	Efficient delivery of client sensory requests while conserving the energy consumption of the sensing devices.	Web Application in Java, Tiny OS 2.2	Back-end server Application runs on Java	WIFI Bluetooth	Base Station
J	Lawson and Ramaswamy et al. [38])	i) Energy Efficient Throttling Algorithm ii) Dynamic Scaling Technique	The determination of tradeoffs between energy efficiency and sensed data quality in the IoT sensing device(s) enabled cloud.	Optimal realization of reliable sensor nodes for the provisioning of desired and quality sensed data while conserving the energy of sensing devices.	Not Specified	Not Specified	Bluetooth 4G	Mobile Phones
K	Pham et al. [55]	i) Queuing (First-In-First-Out) Algorithm	Untimely search for the availability of car parking space	Efficient search and the provisioning of parking space information to car owners.	SIMAN Processor and Simulation Language	Arena Simulation	Zigbee GPRS WIFI	Arduino Local Server
L	Qiang et al. [57]	i) The Multilevel Intelligent Temperature Control Technique	The complexity of regulating minimizing temperature in data center premises	Drastically reduced temperature rate in the cloud data center.	N/A	N/A	4G Zigbee	Base Station
M	Atif et al. [7]	i) Sensed Data Aggregator Technique	The issue of finding available car parking spaces	Optimal provisioning of parking space information to car owners.	N/A	N/A	WIFI	Not Specified

Table 4: The deployment of algorithmic techniques for CloudIoT applications in environmental domain (cont.)

S/N	Authors Name/Year of Publication	Algorithm	Challenges Resolved	Results	Programming Lang.	Simulation Package	Network Comm. Channel	Gateways
N	Dinh and Kim [22]	i) Application Request Aggregator Technique ii) Sensing Low power Listening (SLPL) Technique iii) Sensed Data Aggregator Technique	Limited network bandwidth rate, sensed data redundancy and high energy consumption of sensing devices	Optimal and reliable bandwidth rate for the transmission of relevant sensed data from the sensing devices to the cloud.	TinyOS LPL	Closest-fit-Pattern Matching (CPM) From Mayer Library – Stanford University	WIFI	N/A
O	Yu et al. [81]	i) Best First Algorithm	Not Specified	Not Specified	MATLAB	N/A	WIFI Zigbee	Local Agent Server
P	Bacelo et al. [8]	i) Efficient Linear Algorithm	Sensing service distribution, considering energy consumption	Optimal selection of the shortest route path for obtaining sensing service resulting in energy consumption reduction.	Java Object-Oriented	Not Specified	WIFI 4G Bluetooth	N/A
Q	Li et al. [39]	i) Gabor feature Technique ii) Center-symmetric Local Gabor Binary Pattern (CS-LBP) feature Extraction Technique	The issue of distortion on the sensory image capturing data	Optimal reduction of distortion by retrieving relevant image feature to be used by end users.	MATLAB 2015b	MyEclipse Professional 2014	WIFI	Client Server
R	Chatterjee and Misra [18]	i) Internal Catching Technique ii) External Catching Technique	Sensed data redundancy and transmission rate problem	Sensed data was aggregated and transmission of sensed data from sensor nodes to the cloud efficiently.	C++, Python	TOSSIM Simulator	WIFI	N/A
S	Dinh et al. [22]	i) On-demand Location-based Scheduling Technique	The issue scheduling of sensory nodes for provisioning of on-demand request	Optimal conservation of energy usage and prolong the network lifetime of the IoT sensing devices.	C++	TOSSIM Simulator	4G WIFI	N/A

Table 4: The deployment of algorithmic techniques for CloudIoT applications in environmental domain (cont.)

S/N	Authors Name/Year of Publication	Algorithm	Challenges Resolved	Results	Programming Lang.	Simulation Package	Network Comm. Channel	Gateways
T	Wang et al. [76]	i) Truncation-Point-Optimization Resource Allocation Algorithm ii) QuadTree Decomposition Algorithm	The determination of crucial premium data and irrelevant data at the cloud-edge	Improves prioritization of quality and desired sensed data delivery and experience with minimal energy usage by the IoT sensing devices.	Java Object Oriented	Not Specified	WIFI 4G Bluetooth	EDGE/Fog
U	Qin et al. [58]	i) Geographical Disaster Monitoring and Post-Disaster Rescue System	The issue of inaccurate prediction of earthquakes and landslides that stems from sensed data redundancy	Reduces data redundancy for the efficient and accurate prediction of disasters and the provisioning of timely information to rescue victims after the occurrence of disasters.	Java Object Oriented and Scripting	N/A	GPRS WIFI Bluetooth	Local Server

Discrete-event simulation is performed with the aid of computer systems, virtual simulation software packages, and programming languages. The virtual simulation software provides for sampling from the finite collection of continuous and discrete probability distributions [71]. This is actualized by sampling algorithms that are developed with various programming languages, which are characterized by bounded-mean computing times. Consequently, it analyses the algorithms on a wide range of hypothetical distributions compared to other simulations software. Table 1 depicts the simulation software packages, programming languages, and the computer hardware/Operating System specifications deployed to evaluate the performance of the existing algorithms in CloudIoT platform. The simulation software deployed by existing researchers are briefly discussed as follows; Network Simulation Version2 (NS^2): Network simulator version 2 (NS2) is an open-source simulator designed for a discrete event process, directed for networking research [67]. Its major function is the provisioning of significant "support for simulation of Transmission Control Protocol (TCP), routing and multicast protocols over wired and wireless networks" [51]. It provides a user-friendly Graphics Interfaced design tool for the designing and simulation of a network. It can also be adapted for distributed and parallel simulation as it provides the functionalities of emulation. It provides support for C++ and OTEL programming Language. Network Simulation Version3 (NS^3): It is an open source simulation software package invented to enhance the performance and resolve the constraint limited memory and simulation run time issues in network simulation version2. NS^3 is majorly used to simulate Wireless/IP networks that are made up of WI-FI, WiMAX and LTE deployed for static or dynamic routing protocols. It provides the functionality of a real-time scheduler that enables a variety of simulation-in-the-hoop" uses cases for interacting with real systems.

CloudSim: CloudSim is an open source simulation software with an extensible simulation toolkit that provides the platform for modeling and simulation of cloud computing systems, as well as application provisioning environments [27]. Its virtualization feature provides the possibility of outsourcing the processing of data from IoT device(s) to the cloud server that has almost unlimited resources compared to sensors ([9] and [16]). The major tools used by developers in CloudSim simulator are the Datacenter, Virtual Machines, and Cloudlets. It also supports Java, C++, Python and MATLAB programming Languages.

NetBeans (8.0): It is an integrated development environment that provides computing facilities to software developers and programmers. It is an open source simulation environment that can be accessed freely online. Programming languages such as

Java, Java ME, C/C++, Java EE/Scripting, PHP are used on its platform for developing both object-oriented and Web-based systems.

NetTopo[2] Simulator: It is a commercial base simulation tool that is made up of two functions namely simulation and visualization deployed for simulating and virtualizing network scenarios such as WSNs. It allows the configuration of various programming languages (e.g. Java, Python, C/C++, and MATLAB) in its environment.

OPNET (Optimized Network Engineering Tools): OPNET is a commercial simulation software originated from Massachusetts Institute of Technology (MIT). It provides an extensive development environment supporting the modeling of distributed systems and communication networks [67]. The modeled system features and performance can be evaluated by performing discrete event simulations. OPNET allows Programming languages such as C and C++ to be utilized for the modeling of a system in its environment.

Arena Simulation: It is a simulation software tool developed in a SIMAN code which is then compiled and run without any need to write programming code [75]. It is designed for simulating various types of real-time systems such as IoT parking system.

OMNeT++: It is an open source (Non-commercial setting) discrete event simulator component-based framework. Commonly used for the simulation of computer networks as well as network queuing systems. It only accepts C++ programming Language to be utilized for modeling a network system in its environment.

OWL-S XPlan Package: It is an open-source composition planner that converts OWL-S and OWL services to equivalent domain description, “which are utilized by the AI planner XPlan and PROMETHEA method to generate quality of service aware service composition sequence" [10].

TOSSIM Simulator: It is an open source framework that simulates a wide-range of TinyOS applications. TinyOS is an operating system that is deployed for Wireless Sensor Network environment. TOSSIM is a discrete event simulator that can replace a packet-level communication component for packet-level simulation or replace a low-level radio chip component for a major precise simulation code [74]. It also supports only two programming languages namely, C++ and Python. Monte Carlo Simulation: It is a simulation deployed mainly to obtain estimates of the solution of mathematical problems by means of random numbers [86]. It is considered a sophisticated modeling tool deployed for the analysis of complex computational systems due to its ability to achieve a closer adherence to reality. It is also used for discrete event and other types of simulation processes.

Amazon Elastic Computing Cloud (EC2): It provides scalable computing capability in the Amazon Web Service (AWS) cloud. Its features comprise virtual computing environments, systems software, and numerous configurations of Central Processing Units (CPUs), memory, storage, and networking ability and virtual networks. It is based on commercial settings.

jMetal: It is a simulator software designed for multi-objective optimization problems with meta-heuristics. It is specifically used to simulate and evaluate the performance of Meta-heuristics algorithms such as particle swarm and ant colony optimization. It has also been deployed in various network applications. It also allows Java-based Tools such as EVA2, ECJ and other programming languages such as the C++.

The simulation software packages and programming languages deployed for designing and simulating CloudIoT systems by previous researchers are mostly based on open source. In simple words, they are available freely online. Figure 10 and 11 depicts the comparison of simulation software packages and programming languages utilization over the period 2012 to 2017. It is observed that Monte Carlo has the highest percentage value, indicating that it is the most utilized simulation software package followed by NS[2] and CloudSim with the same percentage values, as presented in Figure 9. It also shows that NeTopo[2], OPNET, NetBeans, and NS[3] has the lowest percentage usage.

MATLAB happens to be the most utilized programming language followed by C++ and Java object-oriented as indicated in Figure 11 below. It shows that the design specifications of CloudIOT systems are actualized optimally by utilizing any of these three aforementioned programming languages. This is because of their simplicity and less consumption of memory space during the execution process. However, MATLAB is known to consume more memory space than C++ and Java. Consequently, academic researchers are presently considering the use of Python because it has been proven to be the easiest to write code with and utilizes limited memory space during the execution process than MATLAB, C++, and Java-oriented.

7 Cloud Gateways and Network Communication Protocols

The physical components deployed for the actualization of CloudIoT infrastructure comprise of RFIDs, IoT sensor nodes, in Table 1 below. The 4G network protocol is an extension of the 3G (for multimedia data transmission), invented to cover gateways and the cloud platform as depicted in previous subsequent figures above. A network gateway (also called fog or edge) links two or more networks for information sharing. Therefore, gateways are deployed in CloudIoT as a communicating intermediary between the cloud platform and IoT sensing devices, thus, to reduce the communication distance between both layers, in order to enhance communication bandwidth rate. Devices such as mobile phones, controllers (e.g. Raspberry Pi) and remote servers are mostly deployed as gateways in the CloudIoT. Communication between the gateway and the cloud is mostly established via a wireless network communication protocol channel. These communication channels, are comprised of the Third/Fourth Generation (3/4 G), General Packet Radio Services (GPRS), Wireless Fidelity (Wi-Fi) and Extensible Messaging/Presence Protocol (XMPP), as observed in previous researches depicted over a wide range distance with its high-speed bandwidth rate of 100 Mbps and broadband of 1Gbps. Wi-Fi enables the

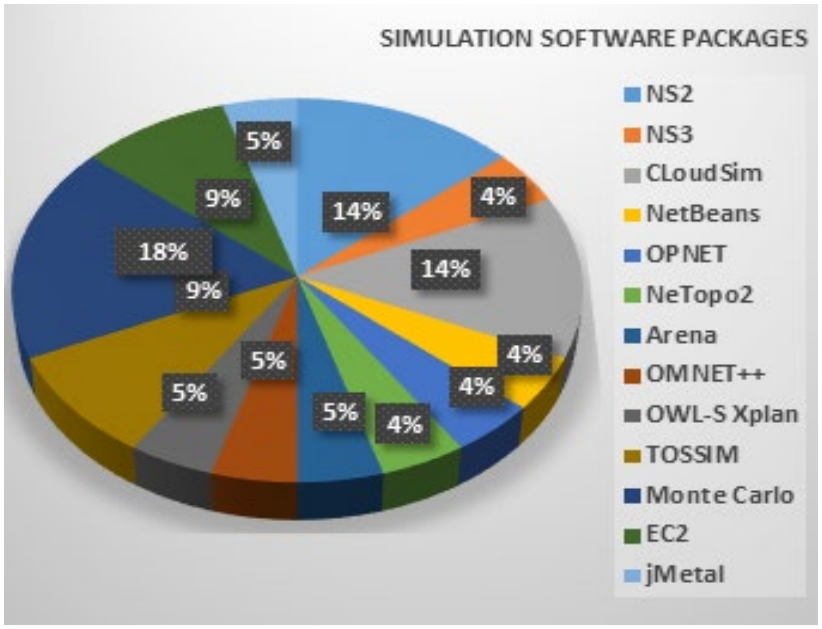


Figure 10: Comparison of CloudIoT simulation software packages

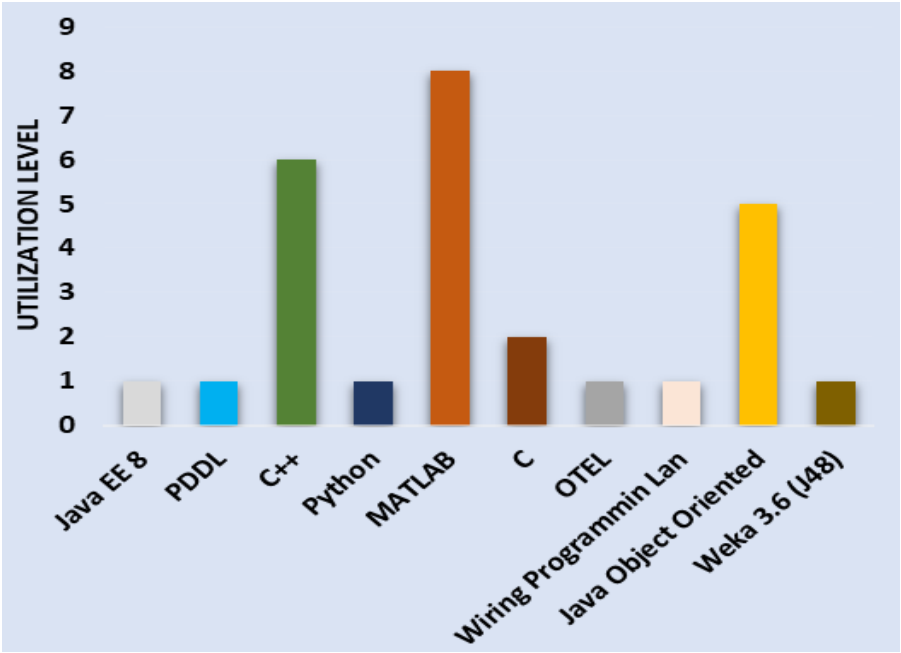


Figure 11: Comparison of programming languages for CloudIoT

interaction between two or more computers and other digital devices that are connected to its network. It utilizes the IEEE 802.11 protocol to transmit data over short distances as well as utilizing high frequencies, operating on either 2.4GHz or 5GHz. XMPP is an open XML-based protocol for transmitting instant messaging and present information on a real-time basis. It was invented by Jabber in the year 1999 as an open-source with its extended features such as Voice over IP and file transfer signaling [52]. GPRS (and SMS) is deployed on a mobile phone network for sending and receiving messages instantly without a dial-up modem connection. Table 5 below shows the features of the aforementioned network communication protocols in terms of their coverage range, bandwidth channel, data/live streaming downlink and uplink speed, frequency band, and network latency. Bandwidth channel is the utilization of lower and upper limit frequency for sending and receiving data over a

network communication protocol. The frequency band is a specific part of the radio spectrum communication frequencies that channels use for the same purpose either for sending or receiving data. On the other hand, network latency is time delayed in the processing and transmission of data or live streaming over a network communication protocol.

The IoT sensing devices and RFID tags receive and transmit data signals to the gateway via a wireless network communication channel such as the Zigbee and Bluetooth. Zigbee is a standards-based wireless technology developed to enable low-cost, low-power wireless device(s) networks for communication [73]. It is specifically designed to control sensor nodes based on the IEEE 802.15.4 standard. Bluetooth is a wireless communication protocol standard developed in the year 1994 for dispatching and receiving data over short distances. Operating over the unlicensed, universally available frequency of 2.4 GHz, with the capability of linking digital devices within a range of 10m at the speed of 1 Mbps [68]. Consequently, increasing its transmitting power directly expands its coverage within a range of 100m.

8 Future Research Direction from Previous Research

Barcelo et al. [8] tend to develop the proposed solution in a cloud computing testbed and formulate an approximation algorithm to improve the computational complexity. Jutila [32] tends to evaluate the adaptive traffic management methods and testing its scalability in a large-scale environment, as well as combining different algorithms to optimize the performance of the IoT applications. Luo and Ren [44] to enhance the application domain of the existing system and applying it to other research disciplines. Kim [33] tends to address the quality aware operations of the proposed system by enhancing its polynomial time algorithm performance. Yang and Shen [80] to formulate the dynamic SS problems that will lead to the design of intelligent optimization algorithms and the development of typical IoT applications. Sareen et al. [65] intend to enhance the efficiency of the existing system by estimating the missing data for training purposes. Li [40] to enhance the recognition speed of the CS-LGBP algorithm to support the implementation of the online real-time recognition system. Yang et al. [80] to improve the visualization of sensing devices of assembly processes in order to facilitate geo-distributed collaborations in real time basis. Narman [48] to investigate the effectiveness of heterogeneity levels of multiple servers and the IoTs requests tasks on cloud computing performance by utilizing the Gini Index.

Kumrai et al. [36] intend to conduct large-scale experiments with standard cloud simulation packages such as the CloudSim, GreenCloud, and ICan-Cloud to resolve the computational complexity of the Multi-objective Particle Swarm Optimization Algorithm (MOPSP). Yu et al. [81] tend to investigate useful analysis and energy estimation involved in the existing technique deployed in the development of IoT applications for smart buildings. Hossain and Muhammad [29] to develop a HealthIoT monitoring framework that will be tested for data security and notification utilities, as well as developing a test trail with real healthcare personnel and patients. Dinh and Kim [35] envisage examining the model for up-stream sensing event traffic congestion for the packet provisioning latency and reliability based on end users and application requests. Ray [61] plans to enhance the proposed system in the near future by

Table 5: Comparisons of network communication protocols deployed in CloudIoT platform

Wireless Network Communi-cation Protocols	Standard	Range	Bandwidth Channel	Data/Video Streaming Downlink	Data/Video Streaming Uplink	Frequency Band	Network Latency
3G	IMT(Universal Mobile Telecom-munications Service)-2000	Wide Range Coverage	5-20MHz	56Mbps	22Mbps	1.8-2.5GHz	100-500ms
4G	IEEE 802.16e-2005	Wide Range Coverage	5-20MHz	1Gbps	100Mbps	2-8GHz	<100ms
GPRS	(ETSI) European Telecommuni-cations Standards Institute	Wide Range Coverage	2000KHz	114Mbps	20Mbps	850-1900MHz	
Wi-Fi	IEEE 802.11	46m (Indoor and 92m Outdoor)	20-40MHz	600Mbps	248Mps	2.4GHz	150ms
XMPP	IETF (Internet Engineering Task Force)	Wide Range Coverage	N/A	N/A	N/A	N/A	N/A
ZigBee	IEEE 802.15.4	10-70m	2450MHz	250Kbps	140Kbps	2.4GHz	100ms
Bluetooth	IEEE 802.15.1	1-100m		24Mbps	1.4Mbps	2.4GHz	200ms

generating alert messages to predict the presence of the hazard in the form of comfort level, before getting into the environment. Also, to investigate how continuous transmission of big data retrieved from IoT sensing device(s) which causes a huge burden on the memory server residing on the cloud will be culminated to avoid data redundancy. Madria et al. [45], intend to develop an application model that is based on plug-and-play and efficient energy scheduling of multiple applications over wireless sensors. Paul et al. [53] to analyze the error-prone inter-sensor nodes links and their effect on the performance of the proposed algorithms will be addressed in the future. Misra and Chatterjee [18] to consider the issue of quality of service parameters of the sensor-enabled cloud platform, additionally to determine the cost-effectiveness of actualizing the virtualization of sensor node features.

Shi et al. [66] intend to improve the existing algorithm in the cloud-assisted fatigue detection system, by introducing a fusion technique to minimize sensed data redundancy, and to address the issue of security and privacy. Alsmirat et al. [6] to investigate the current system by utilizing a different video streaming technology and to extend the existing framework to secure the video stream communication. Kim [34] plans to develop a novel system architecture that integrates mobile cloud, IoT device(s) and communication protocols to enable the transmission of big data from IoT sensing device(s) to the mobile cloud. Critical issues such as security, privacy, quality of service and quality of experience delivery will be addressed in further research. Abawajy et al. [1] are currently developing the proposed algorithm for managing sensory data generated by a sensor device(s) in the enabled CloudIoT as well as testing it in a real-life scenario. They plan to extend the proposed research work by addressing the privacy and security of data transmission.

6 Conclusion

The multi-connected IoT sensing devices are constrained to the extent that they cannot manage the huge amount of data generated as well as providing their physical features in virtualized form. The cloud computing platform through its elasticity and computational features, enable the efficient management of the data resources and the physical features of the IoT device(s) in a ubiquitous and pervasive manner. In this article we first introduced the origin of IoT and cloud, highlighting the motivation that led to the integration of both technologies. Secondly, we presented a brief discussion on existing related survey and review articles based on CloudIoT infrastructure which motivated the current research study. Thus, a detailed analysis is presented on how the data resources and the physical features of the IoT devices are managed effectively and efficiently by the existing algorithms. The comparison of various programming languages, as well as the simulation software packages deployed, to design and evaluate the performance of these algorithms are presented. The various network communication protocols utilized both for internal and external interaction by the CloudIoT system components were discussed. Consequently, the future work as discussed in previous researches is also presented in this article, which paves the way for future research directions**.**

References

[1] Jemal H. Abawajy and Mohammad Mehedi Hassan, "Federated Internet of Things and Cloud Computing Pervasive Patient Health Monitoring System," *IEEE Communication Magazine*, 55(1):48-53, 2017.

[2] Sherif Abdelwahab, Bechir Hamdaoui, Mohsen Guizani, and Taieb Znati, "A Cloud of Things for Sensing-as-a-Service: Architecture, Algorithms and Use Case," *IEEE Internet of Things Journal*, 3(6):1099-1112, 2016.

[3] Nadjib Aitsaadi, Raouf Boutaba, and Yutaka Takahashi, "Cloudification of the Internet of Things," *Annals of Telecommunications (Springer Link)*, 72(1-2):1-2, 2017.

[4] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, 17(4):2347-2376, 2015.

[5] Afiq Muzakkir Mat Ali, Nazrul Muhaimin Ahmad, and Anang Hudaya Muhamad Amin, "Cloudlet-Based Cyber Foraging Framework for Distributed Video Surveillance Provisioning," IEEE 4th World Congress on Information and Communication Technologies (WICT), pp. 199–204, 2014.

[6] Mohammad A. Alsmirat, Yaser Jararweh, Islam Obaidat, and Brij B. Gupta, "Internet of Surveillance: A Cloud Supported Large-Scale Wireless Surveillance System," *The Journal of Supercomputing (Springer Link)*, 73(3):973-992, 2017.

[7] Yacine Atif, Jianguo Ding, and Manfred A. Jeusfeld, "Internet of Things Approaches to Cloud-Based Smart Car Parking, *Procedia Computer Science (Elsevier)*, (96):193-198, 2016.

[8] Marc Barcelo, Alejandro Correa, Jaime Liorca, Anttonia M. Tulino, Jose Lopez Vicario, and Antoni Morell, "IoT-Cloud Service Optimization in Next-Generation Smart Environments," *IEEE Journal on Selected Area in Communications*, 34(12):4077-4090, 2016.

[9] S. Berrahal, N. Boudriga, and A. Bagula, "Cooperative Sensor-Clouds for Public Safety Services in Infrastructure-Less Areas," Paper presented at the Conference on Communications (APCC), 22nd Asia-Pacific, pp. 222-229, 2016.

[10] S. Bharath Bhushan and Pradeep Reddy CH, "A QoS Aware Cloud Service Composition Algorithm for Geo-Distributed Multi-Cloud Domai," *International Journal of Intelligent Engineering and Systems*, 9(4):147-156, 2016.

[11] Alessio Botta, Walter de Donato, Valerio Persico, and Antonio Pescape, "On the Integration of Cloud Computing and Internet of Things," International Conference on the Future Internet of Things and Cloud, pp. 23-30, 2014.

[12] Alessio Botta, Walter de Donato, Valerio Persico, and Antonio Pescape, "Integration of Cloud Computing and the Internet of Things," A Survey, Future Generation

Computer Systems (Elsevier), (56):684-700, 2016.

[13] Leonardo A. Cament, Francisco J. Galdames, Kevin W. Bowyer, and Claudio A. Perez, "Face Recognition under Pose Variation with Local Gabor Features Enhanced by Active Shape and Statistical Models," *Pattern Recognition (Elsevier)*, (48):3371-3384, 2015.

[14] Everton Cavalcante, Jorge Pereira, Marcelo Pitanga Alves, Pedro Maia, Roniceli Moura, Thais Batista, Flavia C. Delicato, and Paulo F. Pires, "On the Interplay of the Internet of Things and Cloud Computing: A Systematic Mapping Study," *Computer Communications (Elsevier)*, 89-90(2016):17-33, 2016.

[15] Frank Cervone H. "An Overview of Virtual and Cloud Computing," International Library Perspectives (EmeraldInsight), 26(3):162-165, 2010.

[16] N. Chandrakant, A. Bijil, P. Puneeth, P. D. Shenoy, and K. Venugopal, "WSN Integrated Cloud Computing for the Then-Care System (ncs) using Middleware Services," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, (4):2278-3075, 2013.

[17] H. C. Chao, "Internet of Things and Cloud Computing for the Future Internet, in Ubiquitous Intelligence and Computing," Lecture Notes in Computer Science, 2011.

[18] Subarna Chatterjee and Sudip Misra, "Dynamic and Adaptive Data Caching Mechanism for Virtualization within Sensor-Cloud," IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS), pp. 1-6, 2014.

[19] Yeong-Sheng Chen and Yu-Ren Chen, "Context-Oriented Data Acquisition and Integration Platform for the Internet of Things," Conference on Technologies AND Applications of Artificial Intelligence, pp. 103-108, 2012.

[20] Waltenegus Dargie and Christian Poellabauer, Fundamentals of Wireless Sensor Networks: Theory and Practice, John Wiley and Sons limited, pp. 1-16, https://onlinelibrary.wiley.com/doi/book/10.1002/9780470666388, [Accessed Date: 06-02-201], 2010.

[21] Ousmane Diallo, Joel J. P. C Rodrigues, Mbaye Sene, and Jianwei Niu, "Real-Time Query Processing Optimization for Cloud-Based Wireless Body Area Networks," *Information Sciences (Elsevier)*, (284):84-94, 2014.

[22] Than Dinh, Younghan Kim, and Hyukjoon Lee, "A Location-Based Interactive Model of the Internet of Things and Cloud (IoT-Cloud) for Mobile Cloud Computing Applications," *Sensors (MDPI Journals)*, 17(3):1-15, 2017.

[23] Maria Fazio and Antonio Puliafito, "Cloud4sens: A Cloud-Based Architecture for Sensor Controlling and Monitoring," *IEEE Communications Magazine*, 53(3):41-47, 2015.

[24] George S. Fishman, "Discrete-Event Simulation, Modeling, Programming, and Analysis," Springer Series Operation Research, 2013.

[25] Dimitrios Georgakopoulos, Prem Prakash Jayaraman Maria Fazia, Massimo Villari, and Rajiv Ranjan, "Internet of Things and Edge Cloud Computing Roadmap for Manufacturing," *IEEE Cloud Computing*, 3(4):66-73, 2016.

[26] Jose A. Gonzalez-Martinez, Miguel L. Bote-Lorenzo, Eduardo Gomez-Sanchez, and Rafael Cano-Parra, "Cloud Computing and Education: A State-of-the-Art Survay," Computers & Education (Elsevier), 80(2015):132-151, 2015.

[27] Tarun Goyal, Ajit Sigh, and Aakansha Agrawal, CloudSim: A Simulator for Cloud Computing Infrastructure and Modeling," *Procedia Engineering (Elsevier)*, (38):3566-3572, 2012.

[28] Song-Nam Hong and Joogheon Kim, "Joint Coding and Stochastic Data Transmission for Uplink Cloud Radio Access Networks," *IEEE Communications Letters*, 18(9):1619-1622, 2014.

[29] M. Shaminm Hossain and Ghulam Muhammad, "Cloud-Assisted Industrial Internet of Things (IIoT)-Enabled Framework for Health Monitoring," *Computer Networks (Elsevier)*, (101):192-202, 2016.

[30] Muhammad Imran, Abas Md Said, and Halabi Hasbullah, "A Survey of Simulators, Emulators and Testbeds for Wireless Sensor Networks," The 2010 International Symposium on Information Technology, (3):897-902, 2010.

[31] Qi Jing, Athanasios V. Vasilakos, Jiafu Wan, Jingwei, and Dechao Qiu, "Security of the Internet of Things: Perspective and Challenges," *Wireless Networks (Springer Link)*, 20(8):2481-2501, 2014.

[32] Mirjami Jutila, "An Adaptive Edge Router Enabling Internet of Things," *IEEE Internet of Things Journal*, 3(6):1061-1069, 2016.

[33] Joongheon Kim, "Energy-Efficient Dynamic Packet Downloading for Medical IoT Platforms," *IEEE Transactions on Industrial Informatics*, 11(6):1653-1659, 2015.

[34] Sungwook Kim, "Nested Game-Based Computation offloading Scheme for Mobile Cloud IoT Systems," *EURASIP Journal on Wireless Communications and Networking (Springer Link)*, (1):1-11, 2015.

[35] Younghan Kim and Thanh Dinh, "An Efficient Interactive Model for On-Demand Sensing-As-A-Services of Sensor-Cloud," *MDPI Sensors*, 16(7):1-28, 2017.

[36] Teerawat Kumrai, Kaoru Ota, Mianxiong Dong, Jay Kishigami, and Dan Keun Sung, Multi-Objective Optimization in Cloud Brokering Systems for Connected Internet of Things," *IEEE Internet of Things Journal*, 4(2):404-413, 2017.

[37] Herald Lampesberger, "Technologies for Web and Cloud Service Interaction: A Survey," *Service Oriented Computing and Applications (Springer Link)*, 10(2):71-110, 2016.

[38] Victor Lawson and Lakshmish Ramaswamy, "Data Quality and Energy Management Tradeoffs in Sensor Service Clouds," IEEE International Congress on Big Data, pp. 749-752, 2015.

[39] Chen Li, Wei, Jiaxue Li, and Wei Song, "A Cloud-Based Monitoring System via Face Recognition using Gabor and CS-LBP Feature," *The Journal of Supercomputing*

(Springer Link), 73(4):1532-1546, 2017.

[40] Shancang Li, Li Da Hx, and Shanshan Zhao. "The Internet of Things: A Survey," *Information Systems Frontiers (Springer Link)*, 17(2):243-259, 2015.

[41] Chun-Han Lin, Pi-Cheng Hsiu, and Cheng-Kang Hsieh, "Dynamic Backlight Scaling Optimization: A Cloud-Based Energy-Saving Service for Mobile Streaming Applications," *IEEE Transactions on Computers*, 63(2):335-348, 2014.

[42] Qiang Liu, Yujun Ma, Musaed Alhussein, Yin Zhang, and Limei Peng, "Green Data Center with IoT Sensing and Cloud-Assisted Smart Temperature Control System," *Computer Networks (Elsevier)*, (101):1040-112, 2016.

[43] Chinyao Low, Yahaueh Chen, and Mingchang Wu, "Understanding the Determinants of Cloud Computing Adoption," Industrial Management & Data Systems (Emeraldinsight), 111(7):1006-1023, 2011.

[44] Shiliang Luo and Bin Ren, "The Monitoring and Managing Application of Cloud Computing Based on Internet of Things," *Computer Methods and Programs Biomedicine (Elsevier)*, (130):154-161, 2016.

[45] Sajay Madria, Vimal Kumar, and Rashmi Dalvi, "Sensor Cloud: A Cloud of Virtual Sensors, *IEEE Software*, 31(2):70-77, 2014.

[46] Lucas D. P. Mendes, Joel J. P. C. Rodrigues, Jaime Lioret, and Sandra Sendra, "Cross-Layer Dynamic Admission Control for Cloud-Based Multimedia Sensor Networks," *IEEE Systems Journal*, 8(1):235-146, 2014.

[47] Nathalie Mitton, Symeon Papavassiliou, Antonio Puliafito, and Kishor S. Trivedi, "Combining Cloud and Sensors in a Smart City Environment," *EURASIP Journal on Wireless Communications and Networking (Springer)*, pp. 1-10, 2012.

[48] Husnu S. Narman, Md. Shohrab Hossain, Mohammed Atiquzzaman, and Haiying Shen, "Scheduling Internet of things Applications in Cloud Computing," *Annals of Telecommunications (Springer Link)*, 72(1):79-93, 2017.

[49] V. Natarajan, A. Balasubramanian, S. Mishra, and R. Sridhar, "Security for Energy Constrained RFID System," 4th IEEE World Workshop on Automatic Identification Advanced Technologies (AutoID'05), pp. 1-6, 2005.

[50] Anne H. Ngu, Mario Gutierrez, Vangelis Metsis, Surya Nepal, and Quan Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling Technologies," *IEEE Internet of Things*, 4(1):1-20, 2017.

[51] NSNAM, What is NS-3? https://www.nsnam. org/overview/what-is-ns-3/, [Accessed: 17-05-208], 2017.

[52] Ozur Ozturk, "Introduction to XMPP Protocol and Developing Online Collaboration Applications using Open Software and Libraries," IEEE, https://iee explore.ieee.org/ stamp/stamp.jsp?arnumber=5478530, [Accessed: 23-05-2018], pp. 21-25, 2010.

[53] Henning Paul, Jorg Fliege, and Armin Dekorsy, "In-Network-Processing: Distributed Consensus-Based Linear Estimation," *IEEE Communications Letters*, 17(1):59-62, 2013.

[54] Rene Peinl, Floriaan Hlzschuher, and Florian Pfizer, "Docker Cluster Management for the Cloud-Survey Result and Own Solution," *Journal of Grid Computing (Springer Link)*, 14(2):265-282, 2016.

[55] Than Nam Pham, Ming-Fong Tsai, Duc Binh Nguyen, Chi-RenDow, and Der-Jiunn Deng, "A Cloud-Based Smart-Parking System Based on Internet-of-Things Technologies," *IEEE ACCESS*, (3):1581-1591, 2015.

[56] Andrea Prati, Roboerto Vezzani, Michele Forniciari, and Rita Cucchiara, "Intelligence Video Surveillance as a Service," *Intelligent Multimedia Surveillance (Springer Link)*, pp. 1-6, 2013.

[57] Liu Qiang, Yujun Ma, Musaed Alhussein, Yin Zhang, and Limei Peng, "Green Data Center with IoT Sensing and Cloud-Assisted Smart Temperature Control System", *Computer Networks*, (Elsevier), 101:1040-1112, 2016.

[58] Lele Qin, Shuang Feng, and Hongyi Zhu, "Research on the Technological Architectural Design of Geological Hazard Monitoring and Rescue-after-Disaster System Based on Cloud Computing and Internet of Things," *International Journal of System Assurance Engineering and Management (SpringerLink)*, 8(1):1-12, 2017.

[59] T. Qu, S. P. Lei, Z. Z. Wang, D. X. Nie, X. Chen, and George Q. Huang, "IoT-Based Real-Time Production Logistics Synchronization System under Smart Cloud Manufacturing," *The International Journal of Advanced Manufacturing Technology (Springer Link)*, 84(1-4):147-164, 2016.

[60] Gladys Rama, Report: AWS Market Share Is Triple Azure's -". https://awsinsider.net/articles/2017/08/01/ aws-market-share-3x-azure.aspx, 2017, [Accessed Date: 8-12-2018].

[61] Partha Pratim Ray, "Internet of Things Cloud Enabled MISSENARD Index Measurement for Indoor Occupants," *Elsevier Measurement*, (92):152-165, 2016.

[62] Partha Pratim Ray, "A Survey of IoT Cloud Platforms," *Future Computing and Informatics Journal (Elsevier)*, 1(1):35-46, 2017.

[63] Bhaskar Prasad Rimal, Admela Jukan, Dimitri's Katsaros and Yves Goeleven, "Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach," *Journal of Grid Computing (SpringerLink)*, 9(1):3-26, 2011.

[64] Mehdi Roopaei, Paul Rad, and Kim-Kwang Raymond Choo," "A Cloud of Things in Smart Agriculture: Intelligent Irrigation Monitoring by Thermal Imaging," *IEEE Cloud Computing*, 4(1):10-15, 2017.

[65] Sanjay Sareen, Sandeep K. Sood, and Sunil Kumar Gupta, "IoT-Based Cloud Framework to Control the Ebola virus Outbreak," *Journal of Ambient Intelligence and Human Computing* (Springer Link) (12):1-18, 2016.

[66] Xiaobo Shi, Yixue Hao, Delu Zeng, Lu Wang, M. Shamim Hossain, Sk Md Mizanur Rahman, and Abdulhameed Alelaiwi, "Cloud-Assisted Mood Fatigue Detection System," *Mobile Networks and Applications (Springer Link)*, 21(5):744-752, 2016.

[67] Saba Siraj, Ajay Kumar Gupta, and Rinku-Badguja,

"Network Simulation Tools Survey," *International of Advanced Research in Computer and Communication Engineering*, 1(4):201-210, 2012.

[68] N. Sriskanthan, F. Tan, and A. Karande, "Bluetooth Based Home Automation System," *Microprocessors and Microsystems (Elsevier),* 6(6):281-289, 2002.

[69] Roman Staszewski, "What are IoT Sensor Devices?" http://zenseio.com/blog/iot-sensor-devices, [Accessed Date: 04-02-2018], 2016.

[70] Harald Sundmaeker, Patrick Guillemin, Peter Friess, and Sylvie Woelffle, "Vision and Challenges for Realizing the Internet of Things," CERP-IoT-Cluster of European Research Projects on the Internet of Things, pp. 1-230, 2010.

[71] Gupta G. Suraj, Mangesh M. Ghonge, Parag D. Thakare, and P. M. Jawandhiya, "Open-Source Network Simulation Tools: An Overview," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 2(4):1629-1635, 2013.

[72] Shahab Tayeb, Shahram Latifi, and Yoohwan Kim, "A survey on IoT Communication and Computation Frameworks: An Industrial Perspective," IEEE 7th Annual Computing and Communication Workshop and Conference, pp. 1-8, 2017.

[73] TechTarget, IoT Agenda, https://internetofthings agenda.techtarget.com/definition/ZigBee, [Accessed: 22-05-2018], 2018.

[74] TinyOS, TOSSIM, http://tinyos.stanford.edu/ tinyos-wiki/index.php/TOSSIM, [Accessed: 22-05-2018], 2013.

[75] Antonio Vieira, Luis Dias, Guilherme Pereira, and Jose Oliveira, "Comparison of Simo and Arena Simulation Tools," https://repositorium. sdum.uminho.pt/handle/ 1822/36949, [Accessed: 23-05-2018], pp. 1-9, 2014.

[76] Wei Wang, Qin Wang; and Kazem Sohraby, "Multimedia Sensing as a Service (MSaaS): Exploring Resources Saving Potentials of Cloud-Edge IoT and Fogs," *IEEE Internet of Things of Journal*, 4(2):487-495, 2017.

[77] Yanzhi Wang, Xue Lin, and Massoud Pedram, "A Nested Two Stage Game-Based Optimization Framework in Mobile Cloud Computing System," IEEE Seventh International Symposium on Service-Oriented System Engineering, pp. 494-502, 2013.

[78] Matt Weinberger, *Amazon Web Services,* Amazon's $18 billion cloud business continues to crush Microsoft and Google — here's the latest scorecard for the cloud war, https://www.pulse.ng/bi/tech/tech-amazons-18-billion-cloud-business-continues-to-crush-microsoft-and-google-heres-the-latest-scorecard-for-the-cloud-war-amzn-id7516962.html*, [Accessed Date: 8-12-2018]*, 2017.

[79] Chen Yang, Shulin Lan, Weiming Shen, George Q. Hung, Xianbin Wang, and Tingyu Lin, "Towards Product Customization and Personalization in IoT-Enabled Cloud Manufacturing," *Clustering Computing (Springer Link)*, 20(2):1717-1730, 2017.

[80] Chen Yang, Weiming Shen, Tingyu Lin, and Xianbin Wang, IoT-Enabled Dynamic Service Selection Across Multiple Manufacturing Clouds," *Manufacturing Letters (Elsevier)*, (7):22-25, 2016.

[81] Jaehak Yu, Marie Kim, Hyo-chan Bang, Sang-Hyun Bae, and Se-Jin Kim, "IoT as Applications: Cloud-Based Building Management System for the Internet of Things," *Multimedia Tools and Applications (Springer Link)*, 75(22):14583-14596, 2016.

[82] Arkady Zaslavsky, Charith Perera, and Dimitrios Georgakopoulos, "Sensing as a Service and Big Data," *Proceedings of the International Conference on Advances in Cloud Computing (ACC)*, pp. 1-6, 2013.

[83] J. Zhou, T. Leppanen, E. Harjula, M. Ylianttila, T. Ojala, C. Yu, and H. Jin, "Cloudthings: A Common Architecture for Integrating the Internet of Things with Cloud Computing, *Proceedings of IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pp. 651-657, 2013.

[84] Chunsheng Zhu, Victor C. M. Leung, Laurence T. Yang, Xiping Hu, and Lei Shu, "Collaborative Location-Based Sleep Scheduling to Integrate Wireless Sensor Networks with Mobile Cloud Computing," IEEE Globecom Workshops - Cloud Computing Systems, Networks and Applications (GC Wkshps), pp. 452-457, 2013.

[85] Chunsheng Zhu, Zhengguo Sheng, Victor C. M. Leung, Lei Shu, and Laurence T. Yang, "Toward Offering More Useful Data Reliability to Mobile Cloud from Wireless Sensor Network," *IEEE Transactions on Emerging Topics in Computing*, 3(1):84-94, 2015.

[86] Eric Zio, *The Monte Carlo Simulation Method for System Reliability and Risk Analysis*, Springer, pp. 1-204, 2013.

Edje Efetobor Abel (Photo not available) is currently a Ph.D. scholar at the Faculty of Computing, Universiti Teknologi Malaysia. He obtained his BSc (Network Computing) in 2009 and MSc (Information Systems Management) in 2010, at Brunel University, West London, United Kingdom. He is also a Lecturer in Delta State University (Delsu) Abraka, Delta State, Nigeira. His area of interests is cloud internet of things, grid computing, network computing, and information systems management.

Muhammad Shafie Bin ABD Latiff (photo not available) received his Ph.D. degree in 2000 from Bradford University, United Kingdom. He is a Professor and currently the Head of Pervasive Computing Research Group at the Faculty of Computing, Universiti Teknologi Malaysia (UTM). His research interests are in computer networks with focus generally on routing protocol, grid, and cloud computing as well as wireless sensor networks.

A Comparative Study of Modified PSO Algorithm and Traditional PSO and GA in Solving University Course Timetable Problem

Paulus Mudjihartono*, Thitipong Tanprasert*, and Rachsuda Setthawong*
Assumption University, Bangkok, 10240, THAILAND

Abstract

University Course Timetable Problem (UCTP) is a challenging NP-complete class as a decision problem that researchers have addressed by introducing many metaheuristic algorithms. Despite the capability of traditional algorithms like Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) Algorithm in solving UCTP, their performance in terms of gain of fitness value (gain) and speed are limited. This work proposes the Modified Abandoned-Reborn Particle Swarm Optimization (MARPSO), and Parallelized Genetic Algorithm-Modified Abandoned-Reborn Particle Swarm Optimization (Par-GA-MARPSO), which enhances traditional GA and PSO to overcome the stated limitations. The comparative study is performed on the two proposed algorithms and traditional GA and PSO with three real cases of UCTP: small, medium, and large datasets, whose goal is to achieve as many constraints as possible. The experimental results show that MARPSO and Par-GA-MARPSO outperform GA and PSO in all cases in terms of gain and speed, and Par-GA-MARPSO compared to PSO performs in average of 5.928 times of improved speed and 1.078 of improved gain.

Key Words: University course timetable problem, genetic algorithm, particle swarm optimization algorithm, parallelized, fitness value, speed.

1 Introduction

University Course Timetabling Problem (UCTP) is a problem of mapping complexity of massive offered courses or lectures to the limited number of classrooms and timeslots. The problem becomes more difficult when it deals with the teacher preferences. In general, UCTP is an optimization problem with some constraints. The constraints could be drawn from the nature of the timetable itself, for instance, non-sharable classrooms and timeslots. Another constraint considered is preferences of teachers or lecturers. Besides constraints, it has an objective function. Objective function is a function in an optimization problem that describes a problem to be optimized. The function can be maximizing or minimizing a certain value out of the problem. In term of UCTP, it means either minimizing the preferences that fail to be fulfilled in the solution, or maximizing the preferences fulfillments as in [4, 8, 11, 16]. These preferences are then meant to be constraints.

A lecturer preference constraint is made from a lecturer, a class, and a timeslot. For example, a lecturer prefers his/her class to hold on Monday at 10 am. Moreover, a class of a course is taught by a lecturer in a certain classroom and timeslot. A lecturer and a course have many-to-many relationship that materialized in a class. A course may contain several classes and a lecturer can teach many classes. A class in some literatures is called a lecture [16], a course [1], a transaction [2], or a work [11]. In this research, it is named lecture. Table 1 shows the terminology this research commonly used.

Table 1: The common terminology

No	Term	Meaning
1	Course	A subject offered by faculty to the students in a certain semester. It may consist of some classes.
2	Teacher/ Lecturer	A person that takes part of a class of a course. He/she teaches a class or some.
3	Classroom	A room that a class takes place. It could be a normal classroom or a lab.
4	Lab	A classroom equipped with facility of computer stations, printers and server.
5	Timeslot	Duration between two certain time stamps where a class takes place.
6	Lecture	A class of a certain schedule of the semester that happens in a certain classroom taught by a lecturer.

Those lectures are scheduled on a tabular form called timetable. A timetable has a general template people usually use. Table 2 shows the template of the timetable. It consists of x rooms and y timeslots, in other words it has $x \times y$ cells that can hold one lecture each. There are m lectures that the timetable holds such that $m \leq x \times y$.

UCTP is a problem of how lectures to be put into the timetable so that as many constraints as can be are fulfilled. Two classic algorithms that solve UCTP are Brute Force (BF)

Program of Computer Science, Faculty of Science and Technology. Email: p5729431@au.edu, thitipong@scitech.au.edu, rachsuda@scitech.au.edu.

Table 2: A template of timetable.

	room 1	**room 2**	.	.	.	**room *x***
timeslot 1	lecture 1	.	.	.	.	lecture *r*
timeslot 2	lecture 2	.	.	.	.	
.	.					.
.	.					.
.	.					.
.	.					.
.	.					.
timeslot *y*	lecture *t*	.	.	.	.	lecture *m*

(BF) and Greedy. Unfortunately, both algorithms do not perform well, in the term of computation burden. BF and Greedy work by searching all solutions. Alternatively, instead of finding all solutions, a heuristics approach is proposed to quickly obtain solution candidates and guess (verify) if they are good enough [19]. This verifying problem becomes a decision problem where at least it can be verified in polynomial time. Also, the fact that a known NP-complete problem, subset sum can be converted to UCTP by adding some requirements makes the problem fall into NP-complete problems. In this case we can build a verifier (algorithm), which runs in polynomial time, to verify every solution supplied.

UCTP requires a verifier that checks a particular solution candidate, which uses polynomial running time for the worst case. This verifier works in a deterministic way. The trial-error solution that a human usually does is considered a kind of this verifier. It verifies each solution candidate; if it gets unsatisfactory solution, then it tries another candidate. In fact both GA and PSO work this way.

GA is an evolutionary based algorithm that uses evolutional mechanism to improve its current situation. As in [6], a population of solution competes with each other to survive. The evolutionary algorithm had been applied to solve many problems such as many-objectives knapsack problem [10], project scheduling problem [15], optimal power flow [5] and community detection from signed social network [14]. On the other hand, PSO is a population based algorithm. Each solution candidate (particle) knows its own current best fitness value so far in the iteration (personal best). It also knows which candidate possessing the best fitness value among the swarm (all candidates) and how good that value is (global best). In moving its particles, PSO takes into account two aspects, social and personal. Social aspect in a particle movement is shown by considering the global best candidate while personal aspect is admitted by considering the best fitness value of that particle along the history of iteration [7]. It had solved the job-shop scheduling [17], clustered the alumni dataset [18], and VAR optimization in electric power systems [9].

This research designs and develops the improved algorithms for solving UCTP that perform better than traditional GA and PSO. The proposed algorithms are the Modified Abandoned-Reborn PSO (MARPSO) and Parallelized Genetic Algorithm-Modified Particle Swarm Optimization (Par-GA-MARPSO). MARPSO treats some bad and good particles to enhance the speed and gain. Par-GA-MARPSO combines the GA and MARPSO in a parallel fashion. The main challenge of this research work is performance improvement of the proposed algorithm in terms of speed and quality of solution obtained in solving UCTP.

The organization of this paper is as follows. Section 2 points out the related works that other researchers had already done. Section 3 describes the model and constraints of UCTP that underlies the proposed algorithms. Section 4 addresses the problem representation while Sections 5 and 6 describe the proposed algorithms. Section 7 shows the experimental results, and lastly, Section 8 concludes the research.

2 Related Works

UCTP is an ongoing research in the past decade. In general, many similar algorithms have been applied to solve many other problems; all of them are NP-class problems. They are all a solution-search based algorithm that solves the problem by proposing many solutions at a time and refining it right after some evaluations. The improvement is gained by some specific mechanisms and the algorithm stops when a stopping criterion is met.

Y. Lei et al used memetic algorithm with hyper heuristic for examination timetabling problems [13]. Hyper heuristic is a heuristic method searching over heuristic space, where several heuristic methods offer solutions. In this algorithm, a coloring graph is used to create heuristic lists. In the evolutionary approach, the iterative creation of candidates is used to get more feasible solutions in the population whereas the crossover and mutation operators are applied to find potential heuristic lists in the heuristic space (high-level search). The results outperform the compared algorithm. However, some drawbacks still exist. The solution taken from heuristic list is not comparable to other solutions in term of soft constraints.

When the resource of the timetable, such as timeslot is changing after a timetable generated, Can Akkan and Ayla Gülcü proposed a robustness measurement [3]. This measurement is to assure that the late changing timeslot is accommodated yet does not make much of constraint penalties. The problem becomes multi-objective optimization that is solved by a multi-objective GA. The result shows high quality Pareto fronts. However, UCTP with predefined timetable resources does not apply this approach.

Adrianto performed a comparison study between PSO with GA in solving UCTP [2]. These algorithms are common in UCTP anyway. The dataset is taken from Binus University Indonesia which manages the lab assistant teaching activity. The research takes hard constraints such as the proper qualification of the assistant, the work shift restrictions, and the teaching crash avoidance. It also takes some soft constraints such as timeslot variations, the number of the shift in a day, and the amount of teachers' burden. The study shows

that PSO outperforms GA. The constraint violations in PSO are less than those in GA. It is obvious that the improved PSO will be worthwhile studied.

Another work proposed a solution for UCTP by modifying the PSO into Abandoned and Reborn PSO or briefly referred as AR-PSO [16]. Some candidates will be treated differently other than that in traditional PSO. Instead of moving the candidates based on their learning, a few numbers of poor candidates were simply abandoned. Consequently, the algorithm will recreate as many new candidates as the abandoned ones. This mechanism gives a slightly improved result. However, it still undergoes weaknesses of being not quick enough to finish. The term 'quick' refers the notion of a UCTP to obtain an acceptable solution with fewer iterations. According to the investigation, the AR part needs to be advanced to open a chance of getting quicker. In fact, it only contains two constraints; both of them are in the form of teacher's preferences. The AR part actually had already changed in [18] merely for solving clustering and called Clustering-AR-PSO (CARPSO). However, it still deals only with the bad particles. In the proposed work, this AR part has further enhanced and becomes Modified AR which is referred as MARPSO.

3 Constraints in UCTP

A timetable is an arrangement of lectures stored into a 2-dimensional layout of classrooms and timeslots. Lecture is viewed as a combination of a lecturer and a course. A course can be taught by some lecturers and a lecturer can teach some courses. However, one lecture is taught by one lecturer only. Subsequently, a timeslot is a fixed uniform time slice (duration) which lecture takes place. There could be many timeslots in the timetable in each week-day. Lectures naturally occupy classroom and timeslot; in the term of 2-dimensional timetable, they lay in the cell of the timetable. The creation process of complete timetable is then simply viewed as inserting lectures into cells of timeslots and classrooms until no more lectures are left (see Table 2). Cell is viewed as a smallest unit container with fixed position and sequence.

Timetable creation process is subjected to several constraints. There are two types of constraints, which are hard and soft constraints. Hard constraints are mandatory to be fulfilled. They may include the exclusiveness of the classroom and lecturer, the finiteness of the classrooms and timeslots, the discreteness of the classroom, timeslot and lecture. These constraints are handled directly by the program. On the other hand, soft constraints can be the university requirements and the lecturers' preferences. The university requirements are some mandatory conditions from the university regulation while the lecturers' preferences are some choices made by lecturers. This kind of constraints becomes an issue of the optimization problem.

Therefore, the constraints should be handled in the optimization problem using university requirements and lecturers' preferences. These constraints are obtained from the quality standard document of the university and lecturers' note survey on their preferences. In this work, university requirements are as follows: (1) a lecturer can teach at most three timeslots a day; (2) a lecturer can only teach at least three days a week (some exceptions apply to some lecturers who undergo special university assignments). On the other hand, lecturers' preferences are as following items: (1) a lecturer may prefer some timeslot of the certain days; (2) a lecturer may avoid some timeslots of the certain days; (3) some lectures may effectively be taught in the certain classrooms. In UCTP, every solution candidate that fulfills these constraints will get rewarded. The more fulfillments it makes, the bigger reward it takes. Therefore, the objective of this UCTP is to maximize the reward.

The constraints dataset is simply the list of the constraints with known keywords by program. There are three columns that form the dataset. Table 3 gives the example of the constraint dataset, where it stores the lecturer or lecture in the first column, preference in the second column, and timeslot or classroom in the last column. The preference (*avoids*, *likes*, *suits*) defines what is in the first and last column. If the preference is *avoids* or *likes* then the first and last column should be a lecturer and timeslot respectively. On the other hand, preference of *suits* will make the first column a lecture and the last column a classroom. This makes the constraints easy to be captured in program logic. For example, the first row of the dataset can be translated as Joko (a lecturer) prefers not to teach at timeslot 16.

Table 3: Constraints dataset

Lecturer_Course	Preference	Timeslot_Classroom
Joko	avoids	16
Devi	avoids	8
Martin	likes	10
Database A Lab	suits	H
. . .	. . .	. . .
OOP E Lab	suits	H

4 Problem Representation

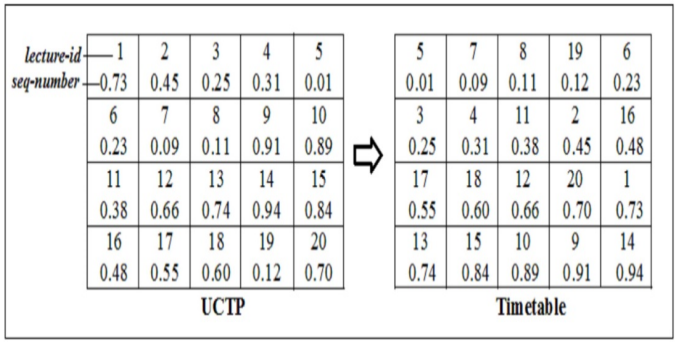
PSO and GA generally use 1-dimensional list (list) to represent the particle and chromosome respectively. A particle is term of PSO while chromosome is of GA. Both are equivalent which represent a solution timetable in the real world. Furthermore, as to limit the problem, there are some requirements to build the timetable which are predefined lecture (lecturer-course relationship), timeslots, and classrooms. Also, the generated timetable is fixed for the whole semester. This means that the timetable will not change during the class activities.

In this research instead of using a list to represent timetable, a matrix will be used. A matrix has an exact structure as in a timetable; the structure of the cells as described in Table 2 can be mapped onto 2-dimensional matrix. Since this matrix representation is different from the list one, it also has different operations upon it. The GA operators performed through this kind of representation need to be modified, which are primary row and/or column wise operations. The row (column) wise operations are the operations that perceived a row (column) as a parameter to be executed.

There are two types of matrix newly generated, which are applicable to all algorithms in this work: UCTP matrix and Timetable matrix. For UCTP matrix, each cell has the information of lecture and its location (index) relative to other cells. In particular, it stores a pair of *lecture-id* (integer) and sequential-number (*seq-number*, floating point). These two numbers are tight on a lecture. The UCTP matrix generation process begins with the creation of *lecture-id,* which is ordered integer number inserted from the top-left cell to the bottom-right cell of the matrix. The *seq-number* is then randomly created and inserted into all the cells. Finally, UCTP matrix is created from the matrix with *lecture-id* ordered. Furthermore, in this condition of the *lecture-id* ordered, the list of the *seq-number* forms the chromosome, and hence all GA operations apply only to UCTP matrix.

On the contrary, the Timetable matrix is formed when its *seq-number* ordered. As a result, the *lecture-id* eventually indicates the lecture location in the Timetable. Therefore, mapping UCTP to Timetable or Timetable to UCTP is done simply by sorting its *seq-number* or *lecture-id*. Figure 1 shows the UCTP and Timetable matrices as a result of the mapping mechanism.

Figure 1: A 2-dimensional structure (matrix) represented UCTP and timetable



5 MARPSO Algorithm

This work proposes Modified Abandoned-Reborn PSO algorithm (MARPSO), which gets rid of the weak point of Abandoned-Reborn PSO algorithm (ARPSO) in [16]. It is suggested in the previous work that ARPSO works in the UCTP. According to our investigation, it has some drawbacks of getting slower. In a nutshell, the idea of performance improvement is that instead of abandoning all considered bad particles as in [16], MARPSO abandons only some bad particles and make others more social than before. Moreover, the proposed algorithm changes the characteristic of some good particles into more selfish. Pseudo code of MARPSO is provided in Algorithm 1 which includes Function 1, 2, 3 and 4.

Before demonstrating the algorithm, it is required to explain the variables and indices used in it. Table 4 shows the variables and indices in the MARPSO. Some of them are also used in Par-GA-MARPSO.

Table 4: Variable and indices used in MARPSO

No	Variable, Index	Meaning
1	t	on t^{th} iteration
2	t+1	on $(t+1)^{th}$ iteration
3	i	i^{th} particle
4	x	position of a particle
5	x_{pb}	personal best position of a particle
6	x_{gb}	global best position of a particle
7	v	velocity of a particle
8	c_1	exploitative constant. It defines the individual behavior of a particle.
9	c_2	explorative constant. It defines the social behavior of a particle.
10	r_1,r_2	randomized random, float (0<r<1)
11	ω	inertia weight
12	μ	fraction to define the working MAR area between global and worst
13	ρ_1	constant to limit the closeness of the global best and worst particle
14	ρ_2	constant to limit the global worst to be considered bad

Algorithm 1: Modified abandoned-reborn PSO (MARPSO).

// Initialization in steps 1 to 5

1. S = **new** Swarm(**new** Particle (N));
   {Particle x_1, x_2, …, x_N}
2. Set *maxConstraints* as the number of all constraints.
3. **read** Constraints C={c_1,c_2,…,$c_{maxConstraints}$};
4. **for each** particle x_i:

$$C_{Xi}(k)=\begin{cases}1, & c_k(x_i)=true, c_k\in C\\ 0, & Otherwise\end{cases}$$

$$f(x_i)=\frac{\sum_{k=1}^{maxConstraints} c_{xi}(k)}{maxConstraints}$$

   endfor

$$f(globalBest)=\max_{i=1..N} f(x_i)$$

$$f(globalWorst)=\min_{i=1..N} f(x_i)$$

   $x_{gb}=x_i$ *where* $f(x_i)=f(globalBest)$

$$x_{pb_i}=0 \text{ for } 0<i\leq N$$

   $v_{ijk}=0$ *for* $0< i\leq N$, $0<j\leq rowN$, $0<k\leq columnN$

   $x_{ijk}=0$ *for* $0< i\leq N$, $0<j\leq rowN$, $0<k\leq columnN$

5. Set the threshold μ, ρ_1 and ρ_2 of fitness value that defines bad particles.

// Iteration in step 6

6. **while not** *criterionMet* **do**:
   Move(**var** *S*: Swarm)
   endwhile

// Termination in step 7

7. Return the global best particle, x_{gb} as the solution.

Function 1: *Move*(**var** *S*:Swarm)

function *Move*(**var** *S: Swarm*):
delta = f(globalBest) – f(globalWorst)
upperBadValue = f(globalWorst) + μ × delta

$$r = \frac{delta}{maxConstraints}$$

for each particle x_i **do**:

$$v_i(t+1) = \omega v_i(t) + c_1 r_1[x_{pb,i}(t) - x_i(t)] + c_2 r_2[x_{gb}(t) - x_i(t)]$$

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

if $x_i(t+1)$ **not** *globalBest* **then**
 condition1 = $f(x_i(t+1)) \le upperBadValue$
 condition2 = $r > \rho_1$
 condition3 = $(f(globalWorst)/maxConstraints) \le \rho_2$
 if *condition1* **AND** *condition2* **AND** *condition3* **then**

$$P_{abandoned}(x_i(t+1)) = \frac{upperBadValue - f(x_i(t+1))}{upperBadValue - f(globalWorst)} \quad (1)$$

  if *rand*() ≤ $P_{abandoned}$ ($x_i(t+1)$) **then**
   makeReborn($x_i(t+1)$) **else** socialize($x_i(t+1)$)
  endif
 endif
 condition4 = $f(globalBest) - f(x_i(t+1)) < \mu * delta$
 if *condition4* **then**
  makeUnsocialized($x_i(t+1)$)
 endif
endif
endfor

$$f(globalBest) = \max_{i=1..N} f(x_i)$$

$$f(globalWorst) = \min_{i=1..N} f(x_i)$$

endFunction

Variable *condition1* is to make sure that the bad particles should be worse than the *upperBadValue*. This *upperBadValue* variable is the tolerable limit of a particle to be considered bad, any particle whose fitness function is worse than this value is considered bad. They must reside in the gray band of Figure 2. In addition, *condition2* is to make sure that MARPSO is taken only when *globalBest* and *globalWorst* particle are not close enough while *condition3* guarantees that MARPSO is taken only when *globalWorst* particle is actually bad. After each bad particle x is found which is falling into *condition1* and *condition2* and *condition3*, its abandoned probability, $P_{abandoned}(x)$ is then calculated using Equation (1). This probability is used to decide if each bad particle x should be either abandoned and straight to make it reborn (*makeReborn*()) or changed to be more social (*socialize*(x)). To make it more socialized, the algorithm changes the movement of the particle by only considering the social part of the PSO algorithm, so therefore it will head toward the global best position. On the other hand, when each good particle x is needed to be more selfish as in fulfillment of *condition4* which is near the *globalBest*, it changes the heading of the particle through only the individual part of the PSO algorithm. The number of all good particles (except *globalBest*) which falls into *condition4* is of σ top best global particles. They behave more explorative than others through function *makeUnsocialized*(x).

The three functions of *MakeReborn*(x), *Socialize*(x), and *MakeUnsocialized*(x) are given in Function 2, 3, and 4. They are maintained to be simple otherwise they will consume the computation burden. For each particle x there will be only one treatment out of these three functions.

Function 2: *MakeReborn*(x)

function *MakeReborn*(x):
 $x_i(t+1) = x_{gb}(t) + \mu * rand()$
endFunction

Function 3: *Socialize*(x)

function *Socialize*(x):
 $x(t+1) = x(t) + c_2 r_2[x_{gb}(t) - x(t)]$
endFunction

Function 4: *MakeUnsocalized*(x)

function *MakeUnsocialized*(x):
 $x(t+1) = x(t) + c_1 r_1[x_{pb}(t) - x(t)]$
endFunction

Figure 2 describes the movement of the particles in MARPSO algorithm. The band made of *worstGlobal* particle and *upperBadValue* (the best bad particle) becomes the area of computation of deciding if a particle is to be abandoned or socialized. On the other hand, inner gray circle becomes the area of computation of recreating the new born particles or turning unsocialized while the white area is intended to

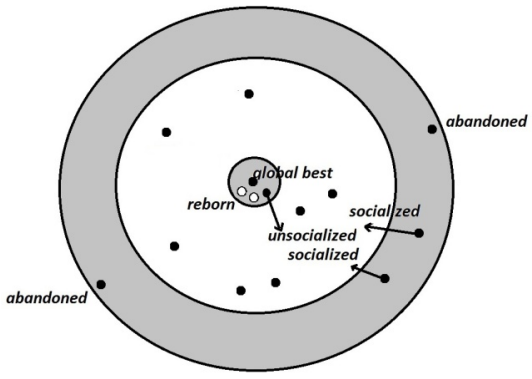


Figure 2: The movement of particles in MARPSO algorithm

become normal PSO computation.

6 Par-GA-MARPSO Algorithm

This work further enhances MARPSO algorithm by combining it with Genetic Algorithm (GA) and parallelism. The idea is from the need for enhancement for GA-MPSO in [17] and makes it even faster. GA-MPSO combines GA and PSO while the Par-GA-MARPSO combines GA and MARPSO. Both run in parallel fashion. In particular, the proposed algorithm utilizes the following operators of GA to move particles in a PSO swarm: mutation (switch and rotation), and crossover.

Par-GA-MARPSO uses the 2-dimension dataset as its particle representation. Therefore, the 2-dimension GA operators are applicable as follows: row-wise mutation, row-wise cross-over and column-wise operations. Figure 3 shows the mechanism of the row-wise mutation operators via switching. The *seq-numbers* of the first row is switched with ones of the third row. Particle (1a) becomes particle (1b), which is a different particle afterward. This happens when the algorithm formulates the dataset in UCTP form. Once it is changed back to Timetable form by sorting the sequence number, it will turn into a different timetable from the previous one.

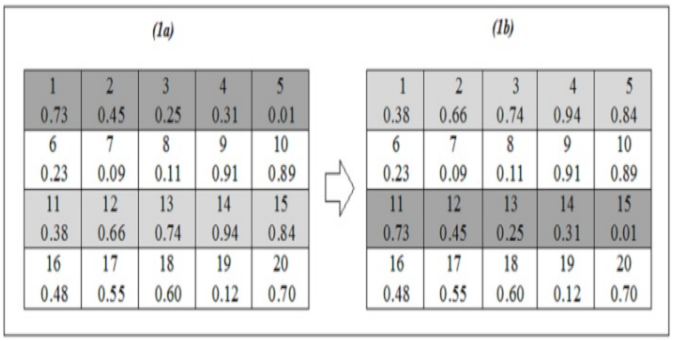


Figure 3: Row-wise mutation by switching rows

Another GA operator used in this mechanism is mutation by rotating the rows. Figure 4 describes such a mechanism. In this figure, function *rotate* (1) is applied to rotate the rows one row up so that the first row becomes the last one; the second one becomes the first row; the third becomes the second and so on. Particle (1a) is row-wise rotated one up and becomes particle (1b). This mechanism guarantees the total changes of the dataset. The result certainly makes the new timetable after transforming from UCTP to timetable form.

Last GA operator is a cross-over that is applied to 2-dimensional fashion. Figure 5 explains how row-wise cross-over happens. Initially, there are two particles: particle (1a) and particle (2a). These two particles are crossed over row wisely. Some rows of particle (1a) are switched with the appropriate rows of particle (2a). As shown in Figure 5, the third and fourth rows of particle (1a) are switched with the third and fourth row of particle (2a). Finally, this operation gives two new particles: particle (1b) and (2b).

The column-wise GA operations do the same fashion; the only difference is that it is done column wisely. The choice between row and column wise operations can be done arbitrarily or by some previous computation. Generally, both row and column fashion operations are considered equally effective.

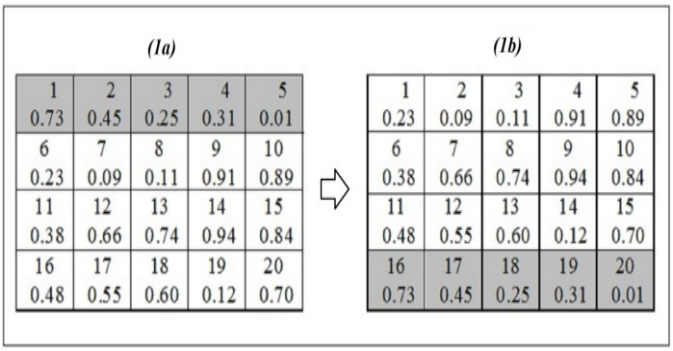


Figure 4: Row-wise mutation by rotating rows

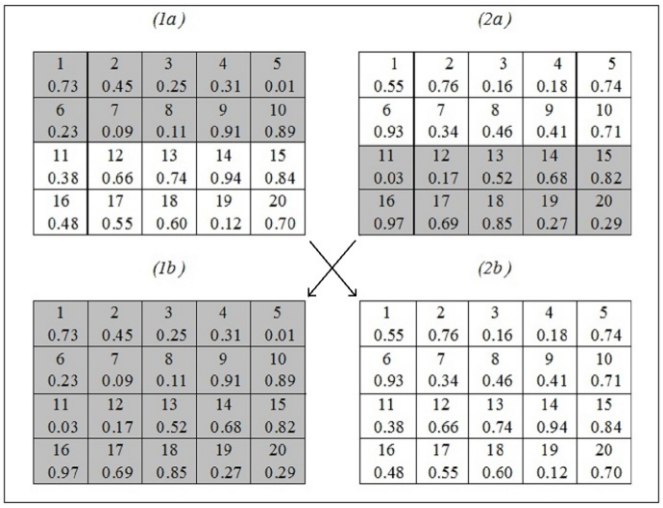


Figure 5: Row-wise cross-over by crossing over the rows

The mutation operation is supposed to change a particle to be a different particle so that it has a greater probability to survive. Hence, it is applied to the weak particles. Consequently, the weaker the particle is, the more likely to be mutated. On the other hand, cross-over is intended to maintain the number of strong particles stays up. A strong particle has a chance to change other particles to have a certain level of similarity with it. A particle that crossed-over with a strong particle has a greater chance to survive. Therefore, the stronger the particle is, the more probability to cross-over with. In UCTP the value of the fitness function determines the likelihood of a particle to be mutated or crossed-over.

In order to enhance the computational speed, a parallel mechanism is applied in moving the particles in the swarm. Using CUDA programming, Par-GA-MARPSO utilizes GPU memory to perform parallel computing. Algorithm 2 which

includes Functions 5, 6, 7, and 8 shows the detail of the algorithm.

Algorithm 2: Parallelized GA-MARPSO

//Initialization in steps 1 to 5

1 S = **new** Swarm(**new** Particle (N));
{Particle $x_1, x_2, \ldots, x_N$}

2 Set *maxConstraints* as the number of all constraints.

3 **read** Constraints C={$c_1,c_2,\ldots,c_{\text{maxConstraints}}$};

4 **for each** particle x_i:

$$C_{Xi}(k)=\begin{cases}1, & c_k(x_i)=true, c_k \in C\\ 0, & Otherwise\end{cases}$$

$$f(x_i)=\frac{\sum_{k=1}^{maxConstraints} c_{xi}(k)}{maxConstraints}$$

endfor

$f(globalBest)=\max_{i=1..N} f(x_i)$

$f(globalWorst)=\min_{i=1..N} f(x_i)$

$x_{gb}=x_i$ *where* $f(x_i)=f(globalBest)$

5 Set the threshold μ, ρ_1 and ρ_2 of fitness value that defines bad particles.

// Iteration in step 6

6 **for** i=1 **to** *maxIteration* **do**:

delta = f(globalBest) – f(globalWorst)

upperBadValue = f(globalWorst) + μ**delta*

r = delta / maxConstraints

for each particle x_i **do concurrently:**

MoveParallel(S)

endfor concurrent

$f(globalBest)=\max_{i=1..N} f(x_i)$

$f(globalWorst)=\min_{i=1..N} f(x_i)$

endfor

//Termination in step 7

7 Stop the iteration and pick the global best one as a solution, x_{gb}.

Function 5: *MoveParallel*(**var** *S*:Swarm)

Function *MoveParallel*(**var** *S*:Swarm):

$$\lambda=\frac{f(x_i)}{f(globalBest)} \quad (2)$$

$x_s = x_{rand()\%maxConstraints+1}$

if λ>*rand()***and** x_s **not** *globalBest* **then**

if *rand()%2=0* **then** *crossoverRowWise*(x_s,x_i)

else *crossoverColumnWise*(x_s,x_i) **endif**

endif

if λ<*rand()* **and** x_i **not** *globalBest* **then**

case *rand()%4* **of**

0: *mutateRowWise*(x_i)

1: *mutateColumnWise*(x_i)

2: *rotateRows*(x_i)

3: *rotateColumns*(x_i)

end case

endif

condition1 = $f(x_i) \le upperBadValue$

condition2 = $r > \rho_1$

condition3 = ($f(globalWorst)/maxConstraints$) ≤ ρ_2

if *condition1* **AND** *condition2* **AND** *condition3* **then**

$$P_{\text{abandoned}}(x_i)=\frac{upperBadValue-f(x_i)}{upperBadValue-f(globalWorst)} \quad (3)$$

if *rand*() ≤ $P_{\text{abandoned}}$ (x_i) **then** *makeRebornGA*(x_i) **else** *socializeGA*(x_i) **endif**

endif

condition4 = $f(globalBest) - f(x_i) < \mu * delta$

if *condition4* **then** *makeUnsocializedGA*(x_i)

endif

endFunction

Like MARPSO algorithm, Par-GA-MARPSO also initiates some similar values. In the initiation part it initiates basic values such as swarm with random particles, constraints read from C dataset, global and worst particle. Particle size is defined from the size of the timetable dataset. The main difference between those two algorithms is that Par-GA-MARPSO uses GA operations on its particle movement and does it concurrently.

In iteration part it needs to compute the probability of being mutated and crossed over. Equation (2) is used to define such requirement which is called λ. The greater λ is, the greater probability to be crossed over is. Besides that the smaller λ is, the smaller probability to be mutated is. This algorithm takes the row-wise and column-wise operations equally into account.

After a particle moves once in the iteration, the algorithm checks if its status is bad particle; if it fulfills the three conditions of *condition1*, *condition2*, and *condition3*, then it will be considered bad particle. A probability value as in Equation (3) is then used to define whether a bad particle should be abandoned and straight to make it reborn or just turned it into more socialized. Another check, *condition4*, is applied to the particle afterward to define if it is near global best and needs to be scattered around by using *makeUnsocializedGA*().

Unlike MARPSO, the three extreme movements in this algorithm are formulated by using the GA operator. Function 6, 7, and 8 show the functions. In these functions less computation is gained by just performing replacement genes of the individual.

Function 6: *MakeRebornGA*(x)

function *MakeRebornGA*(x):

$x_i(t+1) = x_{gb}(t) + \mu * rand()$

endFunction

Function 7: *SocializeGA*(x)

function *SocializeGA*(x):

$x(t+1)$ = crossoverRowWise($x(t)$)

endFunction

Function 8: *MakeUnsocalizedGA*(*x*)

function *MakeUnsocialized*(*x*):
$x(t+1) = mutateRowWise(x(t))$
endFunction

As in [17] the metaheuristic algorithms have the advantageous property which is independent solution to each other. The parallel part is taken care by employing the Compute Unified Device Architecture (CUDA) programming in its main program. CUDA is GPU architecture from NVidia that enables program to run at GPU [12]. This shows that GPU, beside graphics computing, can also do general purpose computing such as those on CPU. It is even better since GPU has parallel property with it. In CUDA programming, the computing works are divided into many smaller works that will be handled by the smallest processor unit called thread. A thread manages its own memory and process concurrently with all other threads. Some threads make one group of threads called block. There could be many blocks with the same specific number of threads in a GPU. Each block shares a memory for communicating among the threads in that block. All blocks are eventually grouped to become a grid; which is used to perform a computation.

There are keywords in CUDA syntax that distinguish what function should run in CPU (Host) and GPU (Device). Keyword `__global__` is located in front of the function definition to indicate that the function is run in device. On the other hand, in calling a function from device, keyword <<<*b*, *t*>>> is used to indicate that the program will use *b* blocks and *t* threads in each block. Furthermore, we need functions placed in the device that also called from the device; here we put keyword `__device__` in front of the function definition.

The particle data structure is a matrix and should be taken care by a thread. The requiring function definitions of *mutateRowWise*, *crossoverRowWise* and *move* are given below.

```
__device__ void mutateRowWise (particle *p,
  curandState *states);
__device__ void crossoverRowWise (particle
  *p1, particle* p2, curandState *states);
__global__ void move (particle *p, particle
  *g, curandState *states);
```

There are more functions that need to be parallelized by the very same manner, such as rotate and mutate and crossover based on column. All these functions are called in the calling function move. The corresponding calling functions are called in *main*() as follows.

```
move <<<BlockSize, ThreadSize>>> (p, g, states)
{
mutateRowWise (p, states);
crossoverRowWise (p, g, states);
}
```

Since we can only access the device data from the device, and the host data from the host, then we need a function that copies data from host to device and vice versa. This function which is called `cudaMemCpy`, helps us accessing data from the adverse source.

A particle or a solution in metaheuristic algorithm is actually independent to each other. Therefore, it can be run independently. This gives us an opportunity to utilize the parallel programming by treating the particles as the independent agents that perform simultaneously. Instead of sequencing the execution of the particle of metaheuristic algorithm, parallelized metaheuristic algorithm executes them in parallel.

Figure 6 shows how the particle is treated in parallel programming. Firstly, the creation of every single particle takes place in the host. Secondly, they are copied into the device so that they can be handled in parallel manner. Thirdly, the parallel execution happens in the device. Finally, the computation result of every particle in the device is copied back to the host [17]. The movement of each particle is taken care by this very fashion.

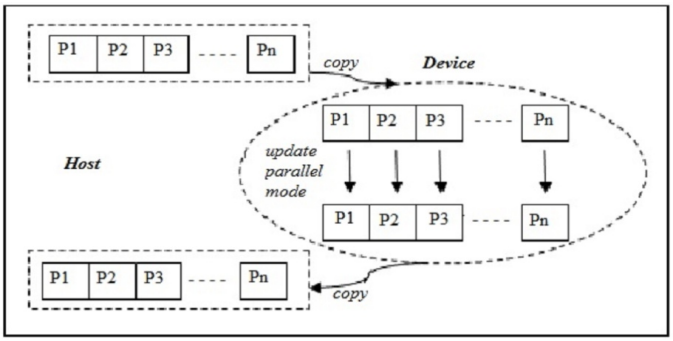


Figure 6: Copy mechanism and parallel performing in the device

7 Experimental Results

This work uses two measurements in order to perform comparative study on performance of two traditional algorithms and the two proposed algorithms, that are goodness or Gain (in the terms of fitness value) and the Speed. We set up 12 experiments, which apply on three different cases (datasets) on the four algorithms. First, the small case is consisting of timetable dataset of 118 lectures, 27 professors, 16 timeslots, 9 rooms (16×9 problem size), and constraints dataset of 74 constraints. Second, the medium case contains timetable dataset of 118 lectures, 27 professors, 20 timeslots, 9 rooms 20×9 problem size and constraints dataset of 86 constraints. Last, the large case is composed of timetable dataset of 145 lectures, 28 professors, 20 timeslots, 12 rooms 20×12 problem size, and constraints dataset of 108 constraints. The experiments need these two datasets, timetable dataset and constraints dataset. Timetable dataset with the shape of

timetable of Table 2 contains predefined lectures, timeslots, and classrooms, only that the position of the lectures in timetable is random. Both timetable and constraints dataset are built in an excel file so they can be fed to the program. The datasets are taken from the real course timetable of study program of informatics of Universitas Atma Jaya Yogyakarta, Indonesia in 2016. All these experiments are implemented under Microsoft Visual Studio 2013 and CUDA 7.5.

Table 5 encompasses the Gain results of all cases. In all cases, the Gain comparisons show that both proposed algorithms MARPSO and Par-GA-MARPSO outperform the traditional GA and PSO. The highlight is the best achievement of Gain made by Par-GA-MARPSO, which is 64.02%, 62.39% and 60.07% of constraints fulfillment of small, medium and large case, respectively. In addition, MARPSO performs better than the traditional GA and PSO with values of 63.18%, 60.59%, and 59.72% for small, medium and large case, respectively.

Table 5: Gain results

No	Algorithm	Gain Results (fraction)		
		Small	Medium	Large case
1	GA	0.5608	0.5614	0.5417
2	PSO	0.5912	0.5728	0.5660
3	MARPSO	0.6318	0.6059	0.5972
4	Par-GA-MARPSO	**0.6402**	**0.6239**	**0.6007**

Another set of experiments is evaluating the speed of all the algorithms by counting the time consumed by each algorithm to get a certain goal of fitness value. The choice of the fitness value is based on the Gain experiment as shown in Table 5. The appropriate value is used corresponding with the performance of the algorithms that it can achieve. The overly value will lead to the infinite loop while the slight value will let the algorithm stop earlier than expected. Therefore, it is chosen the value of 62% for small case, 60% for medium case and 58% for large case.

The results of the second experiment are shown in Table 6. Speed of GA is the worst while speed of PSO much better than speed of GA. Nevertheless the proposed algorithms outperform the traditional ones. The speed comparison between any two algorithms is calculated by dividing the execution time of both algorithms inversely. MARPSO runs faster than PSO by 124.6%, 121.5%, and 121.3% in small, medium and large case, respectively. Finally, the Par-GA-MARPSO is the fastest algorithm. It significantly runs faster than the other three algorithms. Compared to GA and PSO, (GA, PSO), Par-GA-MARPSO has the increasing speed of (5455.1%, 624.3%), (5132.9%, 596.8%) and (4778.8%, 557.2%) for small, medium and large case, respectively.

Regarding the constants (c_1, c_2, μ, ρ_1, ρ_2), MARPSO algorithm needs to be trim to start over to get better results. Until it gets the correct constants, the Gain result is sometimes unsatisfied. In all these cases, constants were chosen

Table 6: Execution time results (in ms)

No	Algorithm	Small case (Gain=0.62)	Medium case (Gain=0.60)	Large case (Gain=0.58)
1	GA	290175.38	349879.38	591013.00
2	PSO	33210.50	40678.13	68908.50
3	MARPSO	26650.88	33466.50	56793.88
4	Par-GA-MARPSO	**5319.38**	**6816.38**	**12367.38**

properly as follows: c_1=1.5, c_2=0.5, μ=0.40, ρ_1=0.15, ρ_2=0.42.

Figure 7 shows the comparison of the Gain results based on Table 5. The trend describes the increasing of the Gain of GA, PSO, MARPSO and Par-GA-MARPSO, consecutively. This confirms that the two proposed algorithms get higher Gain than those of the traditional ones. Compared to GA, MARPSO gets 112.7%, 107.9%, and 110.3% higher in small, medium and large case, respectively. On the other hand, Par-GA-MARPSO in the same measurement has the values of 114.2%, 111.1%, and 110.9% for small, medium and large case, respectively.

Another finding as depicted in Figure 7 shows that the values between cases are consistent. Small case is always gets better Gain than the other cases while the large case has the lowest value of all. All algorithms behave in the same manner in terms of this trend. The algorithms solve the smaller cases better than the larger ones. The problem size and the constraints dataset apparently affect the performance of the algorithm.

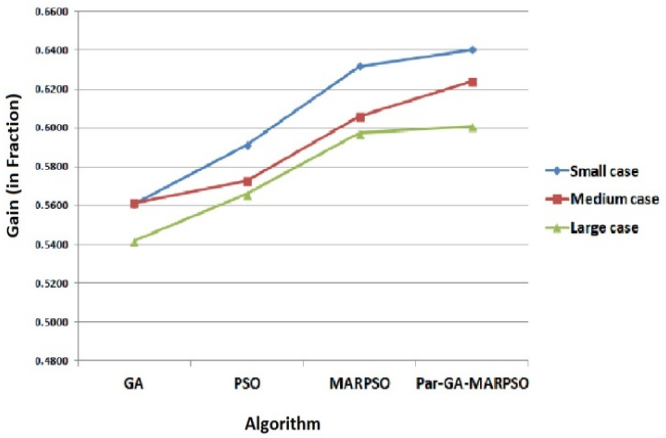


Figure 7: Gain results comparison in fraction

Similarly, the speed aspect is won by the proposed algorithms as shown in Figure 8. The considerable achievement of the speed as expected is made by Par-GA-MARPSO. The parallel feature works well on the three cases. However there are different values between the cases. The small case is executed quicker than other cases. The medium case has the middle value among the cases, and the large case

is the slowest among others. It should be remarked that the parallel algorithm is not entirely parallelized. There is a portion not parallelized. As a result, this portion burdens the algorithm as the problem size grows. The computation of calculating global best and global worst value is the non-parallelized portions.

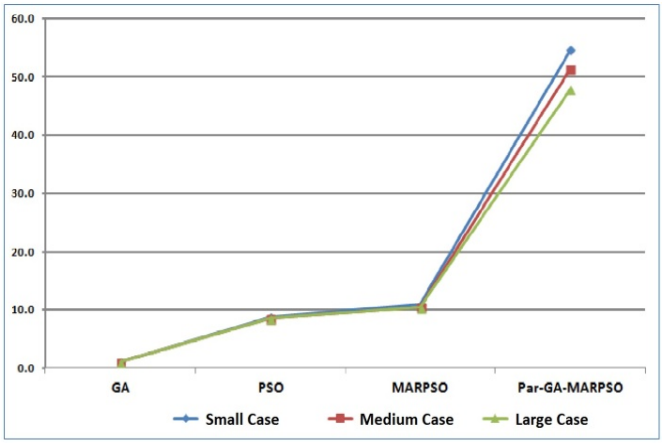


Figure 8: Speed results comparison in scale of GA=1

Result in [2] states that PSO wins over GA while this GA-PSO combination definitely shows the dominant to PSO since MARPSO gets better Gain than PSO and GA as shown in the results of Table 5. Heuristics search over heuristics space like in [13] does not happen in this algorithm. This algorithm simply does heuristics search on real case. Therefore, all solutions are found in one heuristics search level and comparable to each other. Moreover, ARPSO in 1-dimensional representation as in [16] succeeds generating a timetable better than PSO but it runs slower than PSO. It should be stressed also that the MARPSO obviously upgrades the ARPSO and outperforms the PSO both in Gain and Speed.

8 Conclusions

MARPSO outperforms the traditional GA and PSO in all experiments and cases. The better results of Gain and Speed is shown by MARPSO compared to GA than PSO. This also means that PSO is better than GA then the discussion can be contained by PSO comparison only. By evaluating the results, the modification of MARPSO is enhanced by the performance of the PSO. The modification corrects the heading of some bad particles movement. By the same iteration MARPSO gets better Gain, and by the same Gain, MARPSO gets the quicker pace.

In addition, the GA-MARPSO algorithm behavior depends also on the constant selection. The correct constants can lead to the better fitness value however, for some cases they take some more time to finish. Fortunately, the AR mechanism could prevent the algorithm to go further exploring somewhat unnecessary particles, and find better particles quicker. It is proved by the algorithms by giving the better Gain than that in the traditional ones. This at some point will cut the iteration and finish earlier. The crucial issue is that the correct constants of this algorithm cannot be predicted unless we learn them by using some computation, and use it once for all.

The GA, PSO and MARPSO commonly behave in the same manner. They execute the movement of the particles using CPU. However there is a difference between GA and PSO-like algorithm. GA operations lighter than those in PSO-like algorithm since it executes only some particles and the operations itself is merely swapping. Unfortunately, GA iterates the particle movement without storing the global best value while PSO-like algorithm stores the global best value along the iteration. This feature marks the PSO-like algorithm better than GA and consequently makes GA less convergent. Par-GA-MARPSO combines the feature of GA and MARPSO to get a better result of Gain and Speed in all cases.

The parallelized algorithm expectedly grants the fastest run. This proposed algorithm overcomes the lack of speed of the other algorithms especially when compared to GA. This is due to each particle or chromosome is handled concurrently. The concurrent feature is granted by the fact that each particle of the population-based algorithm acts independently. Nevertheless, there are some parts of the algorithm that still need to be run consecutively. These parts earn the burden to the algorithm as they run in a non-parallelized fashion. The next challenges are to minimize the non-parallelized parts and to make them concurrent.

Acknowledgements

The authors would like to acknowledge the support for this research from Ministry of Research, Technology, and Higher Education of the Republic of Indonesia as well as support from Universitas Atma Jaya Yogyakarta, Indonesia.

References

[1] Salwani Abdullah, Hamza Turabieh, Barry McCollum and Paul McMullan, "A Hybrid Metaheuristic Approach to the University Course Timetabling Problem," *Journal of Heuristics*, DOI 10.1007/s10732-010-9154-y, 2012(18):1-23, 2012.

[2] Dennise Adrianto, "Comparison using Particle Swarm Optimization and Genetic Algorithm for Timetable Scheduling," *Journal of Computer Science,* DOI: 10.3844/jcssp.2014.341.346, 10(2):341-346, 2014.

[3] Can Akkan and Ayla Gülcü, "A Bi-Criteria Hybrid Genetic Algorithm with Robustness Objective for the Course Timetabling Problem," *Computers and Operations Research*, DOI: 10.1016/j.cor.2017.09.007, 90:22-32, 2018.

[4] Mohammed Azmi Al-Betar, Ahamad Tajudin Khader and Munir Zaman, "University Course Timetabling using a Hybrid Harmony Search Metaheuristic Algorithm, *IEEE Transactions on Systems, Man, and Cybernetics*,

DOI: 10.1109/TSMCC.2011.2174356, , 42(5):664-681, September 2012.

[5] Emerson Amorim, Shinichiro M Hashimoto, Fabia M Lima, Jose Roberto S Mantovani, "Multi Objective Evolutionary Algorithm Applied to the Optimal Power Flow Problem," *IEEE Latin America Transactions*, DOI: 10.1109/TLA.2010.5538398, 8(3):236-244, June 2010.

[6] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, 2nd Edition, 2015.

[7] Andries P. Engelbrecht, *Computational Intelligence an Introduction*, John Wiley and Sons Ltd, 2nd Edition, 2007.

[8] Cheng Weng Fong, Hishammuddin Asmuni, and Barry McCollum, "A Hybrid Swarm-Based Approach to University Timetabling," *IEEE Transactions on Evolutionary Computation,* DOI: 10.1109/TEVC.2015.2411741, 19(6):870-884, December 2015.

[9] Ying-Yi Hong, Faa-Jeng Lin, Syuan-Yi Chen, Yu-Chun Lin, and Fu-Yuan Hsu, "A Novel Adaptive Elite-Based Particle Swarm Optimization Applied to VAR Optimization in Electric Power Systems," *Mathematical Problems in Engineering*, DOI: 10.1155/2014/761403, 2014(761403):1-14, 2014.

[10] Hisao Ishibuchi, Naoya Akedo, and Yusuke Nojima, "Behavior of Multiobjective Evolutionary Algorithms on Many-Objective Knapsack Problems," *IEEE Transactions on Evolutionary Computation*, DOI: 10.1109/TEVC.2014.2315442, 19(2):264-283, April 2015.

[11] Hamed Jafari and Nasser Salmasi, "Maximizing the Nurses' Preferences in Nurse Scheduling Problem Mathematical Modeling and a Meta-Heuristic Algorithm," *Journal of Industrial Engineering International*, DOI 10.1007/s40092-015-0111-0, 11(3):439-458, 2015.

[12] David B. Kirk and Wen-mei W Hwu. *Programming Massively Parallel Processors: A Hands-on Approach*, Morgan Kauffman, 2010.

[13] Yu Lei, Maoguo Gong, Licheng Jiao, and Yi Zuo, "A Memetic Algorithm Based on Hyper-Heuristic for Examination Timetabling Problems," *International Journal of Intelligent Computing and Cybernetics*, DOI 10.1108/IJICC-02-2015-0005, 8(2):139-151, 2015.

[14] Chenlong Liu, Jing Liu, and Zhongzhou Jiang, "A Multiobjective Evolutionary Algorithm Based on Similarity for Community Detection from Signed Social Networks," *IEEE Transactions on Cybernetics*, DOI: 10.1109/TCYB.2014.2305974 44(12):2274-2287, December 2014.

[15] Leandro L. Minku, Dirk Sudholt, and Xin Yao, "Improved Evolutionary Algorithm Design for the Project Scheduling Problem Based on Runtime Analysis," *IEEE Transactions on Software Engineering*, DOI: 10.1109/TSE.2013.52, 40(1):83-102, January 2014.

[16] Paulus Mudjihartono, "Academic Timetable Generation using Abandoned and Reborn Solution Mechanism of Particle Swarm Optimization," *Proceedings of The Fourth International Conference on Digital Information Processing and Communications*, Kuala Lumpur, Malaysia, pp. 145-151, March 18-20, 2014.

[17] Paulus Mudjihartono, Rachsuda Jiamthapthaksin, and Thitipong Tanprasert, "Parallelized GA-PSO Algorithm for Solving Job Shop Scheduling Problem," *Proceeding of the 2nd International Conference on Science in Information Technology*, Balikpapan, Indonesia, DOI: 10.1109/ICSITech.2016.7852616, pp. 103-108, October 26-27, 2016.

[18] Paulus Mudjihartono, Thitipong Tanprasert, and Rachsuda Jiamthapthaksin, "Clustering Analysis on Alumni Data using Abandoned and Reborn Particle Swarm Optimization," *Proceeding of the 8th International Conference on Knowledge and Smart Technology*, Chiang Mai, Thailand, DOI: 10.1109/KST.2016.7440500, pp. 22-26, 2016.

[19] Michael Sipser, *Introduction to the Theory of Computation*. Boston, MA: Thomson Course Technology, 2nd edition, 2006.

Paulus Mudjihartono is a Ph.D. candidate in Computer Science from the Assumption University of Thailand. His research focus is on metaheuristic algorithms, and parallel computation. Several of his last publications are concerning modified PSO, cluster analysis and the implementation of CUDA parallelism. He also does some review for conferences and journals. He currently works at Universitas Atma Jaya Yogyakarta, Indonesia.

Thitipong Tanprasert is an Assistant Professor in Computer Science and the Dean of the Vincent Mary School of Science and Technology at Assumption University of Thailand. He received his bachelor degree in Electrical Engineering from Chulalongkorn University in 1987, the master and doctor of philosophy in Computer Engineering degrees from University of Louisiana at Lafayette in 1989 and 1993, respectively. His research interests are in neural network computation, data science, and machine learning applications.

Rachsuda Setthawong received a Ph.D. in Computer Science from the University of Houston, Texas, United States. She is an Assistant Professor in the Department of Computer Science and the Graduate Director in Computer Science at the Assumption University. Her area of expertise includes data mining and knowledge discovery, cluster analysis, text mining, social network analysis, optimization algorithms, recommender systems, fuzzy systems, and mobile applications. Her current research focuses on hybrid optimization algorithms, Twitter ranking algorithms, lip reading algorithms, an English-Thai translation framework for non-timing aligned parallel corpora, and opinion mining. She also serves as a reviewer for many conferences and journals.

Index

Authors

A

B

C-G

H-J

K-L

M-O

Key Words

A

B-C

D

E-F

G-H

I-J

K-L

M

N-O

P-Q

Instructions for Authors

The International Journal of Computers and Their Applications is published multiple times a year with the purpose of providing a forum for state-of-the-art developments and research in the theory and design of computers, as well as current innovative activities in the applications of computers. In contrast to other journals, this journal focuses on emerging computer technologies with emphasis on the applicability to real world problems. Current areas of particular interest include, but are not limited to: architecture, networks, intelligent systems, parallel and distributed computing, software and information engineering, and computer applications (e.g., engineering, medicine, business, education, etc.). All papers are subject to peer review before selection.

A. Procedure for Submission of a Technical Paper for Consideration:

1. Email your manuscript to the Editor-in-Chief, Dr. Fred Harris, Jr. Fred.Harris@sce.unr.edu.
2. Illustrations should be high quality (originals unnecessary).
3. Enclose a separate page for (or include in the email message) the preferred author and address for correspondence. Also, please include email, telephone, and fax information should further contact be needed.

B. Manuscript Style:

1. The text should be, **double-spaced** (12 point or larger), **single column** and **single-sided** on 8.5 X 11 inch pages.
2. An informative abstract of 100-250 words should be provided.
3. At least 5 keywords following the abstract describing the paper topics.
4. References (alphabetized by first author) should appear at the end of the paper, as follows: author(s), first initials followed by last name, title in quotation marks, periodical, volume, inclusive page numbers, month and year.
5. Figures should be captioned and referenced.

C. Submission of Accepted Manuscripts:

1. The final complete paper (with abstract, figures, tables, and keywords) satisfying Section B above in **MS Word format** should be submitted to the Editor-in-chief.
2. The submission may be on a CD/DVD, or as an email attachment(s). **The following electronic files should be included:**
   - Paper text (required)
   - Bios (required for each author). Integrate at the end of the paper.
   - Author Photos (jpeg files are required by the printer)
   - Figures, Tables, Illustrations. These may be integrated into the paper text file or provided separately (jpeg, MS Word, PowerPoint, eps). title of the paper.
3. Specify on the CD/DVD label or in the email the word processor and version used, along with the title of the paper.
4. Authors are asked to sign an ISCA copyright form (http://www.isca-hq.org/j-copyright.htm), indicating that they are transferring the copyright to ISCA or declaring the work to be government-sponsored work in the public domain. Also, letters of permission for inclusion of non-original materials are required.

Publication Charges:

After a manuscript has been accepted for publication, the author will be invoiced for publication charges of $50 **USD** per page (in the final IJCA two-column format) to cover part of the cost of publication. For ISCA members, $100 of publication charges will be waived if requested.

January 2014