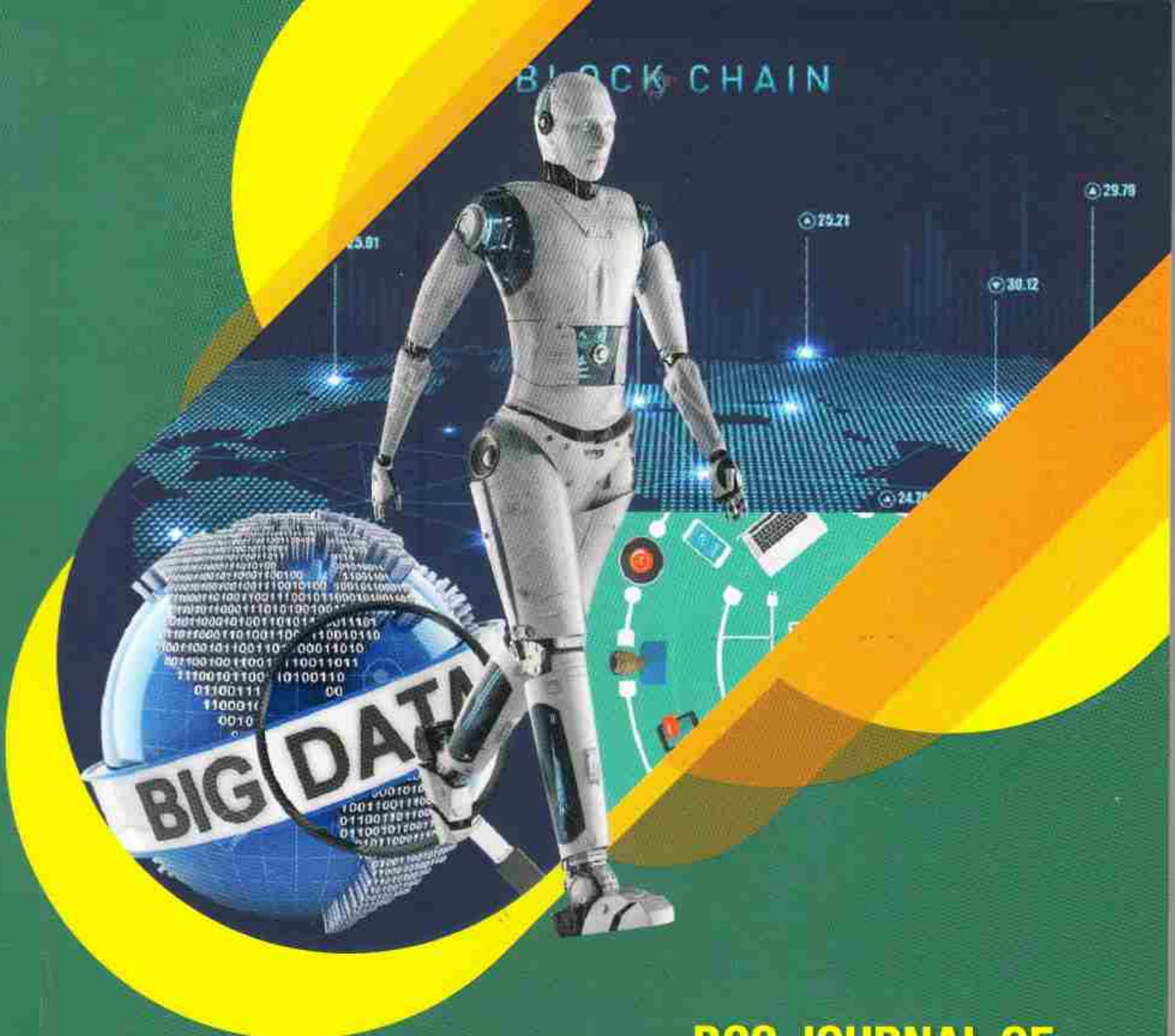


BCS JOURNAL OF COMPUTER AND INFORMATION TECHNOLOGY

Vol. 02, 2020
ISSN: 2664-4592 (Print), 2664-4606 (Online)

বাংলাদেশ কম্পিউটার সোসাইটি
BANGLADESH COMPUTER SOCIETY
E-mail : bcsjcit@gmail.com, Cell: 01753-827949
Web: www.bcsbd.org.bd

BCS Journal of Computer and Information Technology

Vol. 02, 2020
ISSN : 2664-4592 (Print)
2664-4606 (Online)

ADVISORY BOARD

Prof. Dr. Abdul Matin Patwari
Former Vice Chancellor, Bangladesh University of Engineering and Technology, Bangladesh

Prof. Dr. Md. Lutfor Rahman
Vice Chancellor, Daffodil International University, Bangladesh

Late National Prof. Dr. Jamilur Reza Choudhury
Former Vice Chancellor, University of Asia Pacific & BRAC University, Bangladesh

Prof. Dr. Mohammad Kaykobad
Bangladesh University of Engineering and Technology, Bangladesh

বাংলাদেশ কম্পিউটার সোসাইটি
BANGLADESH COMPUTER SOCIETY

BCS Journal of Computer and Information Technology

Vol. 02, 2020
ISSN : 2664-4592 (Print), 2664-4606 (Online)

EDITORIAL BOARD

Prof. Dr. Hafiz Md. Hasan Babu, University of Dhaka	Chief Editor
Prof. Dr. Mohammad Shamsul Arefin, CUET	Associate Editor
Prof. Dr. Mohammad Shorif Uddin, Jahangirnagar University	Associate Editor
Prof. Dr. Lafifa Jamal, University of Dhaka	Associate Editor

MEMBERS

Prof. Dr. A.K.M. Ashikur Rahman, Head, Department of CSE, BUET
Prof. Dr. Md. Mustafizur Rahman, Chairperson, Department of CSE, Dhaka University
Prof. Dr. Asaduzzaman, Head, Department of CSE, CUET
Prof. Dr. Pintu Chandra Shill, Head, Department of CSE, KUET
Prof. Dr. Boshir Ahmed, Head, Department of CSE, RUET
Prof. Dr. Md. Anisur Rahman, Head, CSE Discipline, Khulna University
Prof. Muhammad Anwarul Azim, Chairman, Department of CSE, University of Chittagong
Prof. Dr. Fazlul Hasan Siddiqui, Head, Department of CSE, DUET
Prof. Dr. Bimal Kumar Pramanik, Chairman, Department of CSE, University of Rajshahi
Prof. Dr. Mohammed Nasir Uddin, Chairman, Department of CSE, Jagannath University
Prof. Mohammad Abdullah Al Mumin, Head, Department of CSE, SUST

বাংলাদেশ কম্পিউটার সোসাইটি
BANGLADESH COMPUTER SOCIETY

BCS Journal of Computer and Information Technology

Vol. 02, 2020
ISSN : 2664-4592 (Print), 2664-4606 (Online)

PUBLICATION COMMITTEE

Prof. Dr. Hafiz Md. Hasan Babu
President, Bangladesh Computer Society

Prof. Dr. Mohammad Shorif Uddin
Vice President (Administration), Bangladesh Computer Society

Mr. Sk. Md. Zaminur Rahman
Vice President (Finance), Bangladesh Computer Society

Prof. Dr. Engr. Mohammad Shamsul Arefin
Vice President (Academic), Bangladesh Computer Society

Mr. Md. Tomig Uddin Ahmed
Secretary General, Bangladesh Computer Society

Mr. Md. Akteruzzaman
Joint Secretary (Administration), Bangladesh Computer Society

Engr. Sk. Rawnaqus Salehin
Joint Secretary (Finance), Bangladesh Computer Society

Mr. Rezaul Karim
Joint Secretary (Academic), Bangladesh Computer Society

Prof. Dr. Lafifa Jamal
Treasurer, Bangladesh Computer Society

বাংলাদেশ কম্পিউটার সোসাইটি
BANGLADESH COMPUTER SOCIETY
House#164, 5th floor, Meher Tower, Sonargaon Road, Hatirpool Kacha Bazar, Dhaka-1205
E-mail: bcsjcit@gmail.com, Mobile: 01753827949, Web: https://www.bcsbd.org.bd/

BCS Journal of Computer and Information Technology

Vol. 02, 2020

ISSN: 2664-4592 (Print), 2664-4606 (Online)

Published By:

Bangladesh Computer Society (BCS)

Dhaka-1205, Bangladesh

© All rights reserved

Publish annually. All communication to be made with the chief editor of BCS Journal of Computer and Information Technology, Bangladesh Computer Society, Dhaka-105, Bangladesh.

House#164, 5th floor, Meher Tower, Sonargaon Road, Hatirpool Kacha Bazar, Dhaka-1205.
E-mail: bcsjcit@gmail.com, Mobile: 01753827949, Web: https://www.bcsbd.org.bd/

BCS Journal of Computer and Information Technology

Vol. 02, 2020
ISSN: 2664-4592 (Print), 2664-4606 (Online)

Contents **Page**

BCS Journal of Computer and Information Technology
Vol. 02, 2020, ISSN: 2664-4592 (Print), 2664-4606 (Online)

A New Approach for Reversible Online Testability with No Ancillary Inputs and Garbage Outputs

Hafiz Md. Hasan Babu
Department of Computer Science and Engineering
University of Dhaka, Dhaka-1000, Bangladesh
Email: hafizbabu@du.ac.bd

Abstract: This paper proposes a cost-efficient reversible online testable unit. We have presented an efficient technique for designing the compact reversible online testable unit with the optimum time complexity. We show that the proposed online testable technique has time complexity O(*$log_2 n$*), whereas the best known existing technique has O(*n*), where *n* is the number of inputs in the reversible circuit. In addition, our design proposes three new reversible online testing blocks: An Input Copying Block (ICB) for copying the inputs of the given reversible ESOP (Exclusive-OR Sum of Products) function, an Output Copying Block (OCB) for copying the required outputs of the proposed online testable circuit and a Myriad Feynman Gate Block (MFGB) that acts as the error detection band for checking online testability of the proposed reversible circuit. Moreover, the proposed reversible online testable circuit is the first ever online testable in the literature till now, which is garbage and ancilla free. The comparative study using benchmark functions shows that the proposed online testable circuit performs better than the existing ones in terms of quantum gates, delays, area and power, i.e., on an average, the proposed online testable circuit improves 68.34%, 38.23%, 75.84%, 17.36%and 35.82% in terms of area, power, delay, quantum cost and on the number of transistors, respectively, than the best known existing one.

Keywords: Online testability, Quantum cost, Garbage output, ESOP, Benchmark function.

1. Introduction

Reversible logic is an unprecedented logic by which a reversible circuit can recover the bit loss from its mapping of unique input and output vectors, whereas an irreversible or a conventional logic is unable to recover its inputs from its output patterns or vice versa and thus it losses information. Now-a-days, reversible logic has been immensely used in various applications such as power efficient device, DNA computing, quantum computing [1] and nanotechnology [3]. In 1960s, Landauer [2] vigorously proved that the losing of each bit of information dissipates the significant amount of energy [5] which is at least $KT\times ln2$ joules, where K is the Boltzmann constant and T is the temperature [5] at which the system is operating. As reversible logic has one-to-one correspondence [4, 5] between its input and output vectors, no information is lost approximately in this technique and probably zero power dissipation would be achieved. As such, reversible logic would be a solution in the pave of nano-technological nourishment era [3]. Though reversible circuits consist of reversible gates [4], the performance of a reversible circuit depends on some matrices such as area, power, delay, quantum cost and number of gates etc. Key challenge is to overcome these impediments and optimize these parameters as much as possible [6, 22].

Testing is an important part of reversible logic since it is necessary to make sure that the resulting reversible circuit is fault free. A very few previous research works have been done on testable reversible circuits. In reversible logic, testing can be one of the major problems, as the levels of reversible logic are significantly higher than the standard logic. Testing approaches of reversible circuits are at the beginning of development, and the recent approaches are described in [16-20].

Five main contributions are addressed in this paper are as follows:

- A new reversible online testable technique has been introduced with the optimum time complexity till now in the literature.
- A new reversible input copying block and an output copying block circuits are presented with the fewest number of quantum gates.
- A distinct Feynman gate, namely, MFGB has been introduced with the optimal quantum cost that serves as an error detection band prohibiting garbage outputs from passing as outputs.
- A reversible online testable circuit is presented using the proposed online testable technique, two circuit blocks and a distinct Feynman gate with the optimum number of quantum gates, area, power, delay and transistors with no ancillary inputs and garbage outputs.
- The proposed online testable circuit is also compared with the existing ones in terms of area and power, and with the help of lemmas and theorems, the reversibility and efficiency of the proposed online testable circuit has also been established.

In the next section, we have discussed basic definitions and properties of reversible circuits. In Section III, we have discussed the earlier approaches and their limitations. In Section IV, we have described our motivation towards designing a new reversible online testable circuit. In Section V, we have proposed a fast online testable technique and designed the testable circuit. In Section VI, we have discussed performance analysis of the proposed method. Last of all, we have concluded the paper in Section VII.

2. Basic Definitions of Reversible Gates and Their Properties

In this section, we have represented the basic ideas and definitions of some of the properties of reversible logic.

2.1 Reversible Gate

Reversible gate is an n-input, n-output circuit that produces a unique output pattern [4, 23-24] for each possible input pattern (one-to-one correspondence). For an $n \times n$ reversible gate if the input vector be I_v and the output vector be O_v, then $I_v = (I_1, I_2, I_3 ... I_n)$ and $O_v = (O_1, O_2, O_3 ... O_n)$. The relationship between these vectors is denoted as $(I_v \leftrightarrow O_v)$ [4, 23-24]. A block diagram of an $n \times n$ reversible gate is shown in Fig: 1.

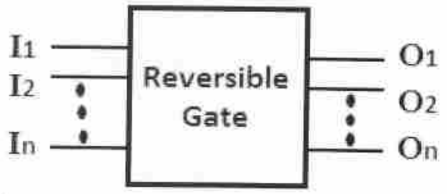


Fig. 1: Block diagram of an $n \times n$ reversible gate

2.2 Reversible Parity Preserving Gate

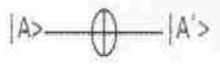
Reversible parity preserving gate is used to detect the error of a reversible gate which constantly preserves the same parity between inputs and outputs. This property is $I_1 \oplus I_2 \oplus ... \oplus I_n = O_1 \oplus O_2 \oplus ... \oplus O_n$ which allows to detect a faulty signal from the circuit's primary outputs [15].When a reversible circuit is implemented using only parity preserving reversible gates, the whole circuit itself preserves the parity and thus it can able to detect the fault [15].

2.3 Online Testability

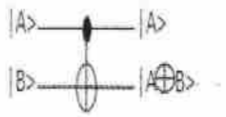
According to [19], there are two types of testing approaches: Online (concurrent testing) and Offline (non-concurrent testing), or both can be combined. ***Online testing approach*** is working while the system is performing its normal operation which allows faults to be detected in real time [24].

2.4 Quantum Cost

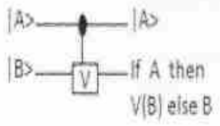
Quantum cost can be computed by substituting the reversible gates of a circuit by a cascade of elementary quantum gates [1, 7]. Elementary quantum gates realize the quantum circuit that are inherently reversible and manipulate qubits rather than pure logic values [10]. Quantum cost of the reversible circuit is the number of basic quantum gates that the circuit consists of [1, 7, 10]. Graphical representation of basic quantum gates, *i.e.*, NOT, CNOT, Controlled-V and Controlled-V^- are shown in Fig.2(a), Fig.2(b), Fig.2(c) and Fig.2(d), respectively.



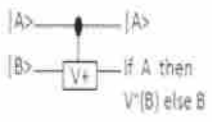
(a) NOT Quantum Gate [10]



(b) CNOT Quantum Gate [10]



(c) Controlled-V Gate [10].



(d) Controlled-V^+ Gate [10].

Fig. 2: Basic quantum gates

2.5 Garbage Outputs and Constant Inputs

An unwanted or unused output of a reversible gate which is not used as the input of other reversible gates, is known as a ***garbage output***. Garbage outputs are needed only to maintain the reversibility. Heavy price is paid off for each garbage output [22]. ***Constant inputs*** are the inputs of a quantum gate (or circuit) that are either set to 0 or 1 [6]. Constant inputs are also known as ***ancillary inputs*** [6].

2.6 Delay

The **delay** of a reversible circuit is proportional to the depth of the circuit [7]. The number of stages in the quantum representation of a reversible circuit is called the **depth** of the circuit [7]. Delay time in a transistor-based circuit is calculated using the formula $Delay_{propagation}=RC$, where C is the total capacitance and R is the effective resistance of the circuit [21]. The delay of an electronic device must be kept minimum for making it faster [7, 21].

3. Background Study of Existing Online Testable Approaches

In this section, we have discussed some previous approaches for constructing online testable reversible circuits. We have also analyzed design issues and performance metrics with benefits and limitations of these approaches.

3.1 Online Testable Approach Using R1, R2 and R Reversible Gates [16]

Vasudevan *et al.* [16] proposed an approach for designing online testable reversible circuits using reversible gates. This approach proposes three new reversible logic gates, namely R1, R2 and R reversible gates. Gate R1 implements arbitrary functions and R2 incorporates online testability features into the circuit. R1 can generate all Boolean functions by setting different values in its input lines. R2 gate is used in the circuit to determine if there is any single bit fault in R1 gate or in itself. If R1 is fault free, the parity output of R1 and the parity output of R2 should be complement to each other, which is tested using a two-pair rail checker. Otherwise, the circuit is assumed to be faulty. This technique is easier to implement for detecting single bit faults. Main disadvantages of this method are as follows: it generates huge garbage outputs, requires a large number of constant inputs as well as area, power and delay to check the online testability. Moreover, faults may occur due to excessive garbage outputs which also makes the final output of the circuit faulty although the circuit is not actually faulty.

3.2 Online Testable Approach Using Testable Reversible Cells [20]

Mahammad *et al.* [20] proposed an approach that converts an existing reversible circuit to an online testable circuit. This approach consists of the following steps: Firstly, it transforms an $n \times n$ reversible circuit (gate) into an $(n+1) \times (n+1)$ reversible circuit (namely, DR_a (deduced reversible gate-a) by adding an additional input and an additional output with the initial circuit (gate). Secondly, it adds an $(n+1) \times (n+1)$ reversible circuit(Deduced reversible gate-b, DR_b) with the previous circuit (DR_a) to calculate the parity of the initial circuit. Thirdly, the reversible circuits DR_a and DR_b are combined together to construct a testable reversible cell (TRC). Lastly, two parity outputs from each TRC cell are combined together to form a test cell(TC) to detect the error of the online testable circuit. This method has less time complexity and it requires fewer quantum gates than [16]. However, it still requires huge delay, quantum cost, area, power, garbage outputs and ancillary inputs.

3.3 Online Testing of ESOP-Based Circuits [18]

Nayeem *et al.* proposed an approach [18] for online testing of an ESOP-based circuit [17]. To convert an ESOP circuit into an online testable circuit, this approach adds some additional NOT and CNOT quantum gates as shown in Fig. 2(a) and Fig. 2(b), respectively. A special representation of a Toffoli gate is the extended Toffoli gate (ETG) [18],which is also used in the paper to replace the conventional Toffoli gates of the ESOP circuit. A parity line is required for detecting a fault which is initialized with a 0 (zero). In the paper, an n-bit Toffoli gate is replaced by an $(n+1)$-bit ETG gate, where the last $(n+1^{st})$ bit of the ETG gate is connected to the parity line. The method works as follows: Firstly, n bits of the ETG gate are same as that of an n-bit Toffoli gate for the function or an expression. Secondly, EXOR operation is performed of each input line with the parity line in the given circuit and thirdly, EXOR operations of both input and output lines are performed with the parity line. This method makes all benchmark functions online testable in an efficient and faster way than existing techniques [16, 20], and reduces gate level delays, but it still requires a large number of garbage outputs and constant inputs which causes faults at the final output of the circuits though the circuits are not actually faulty. Moreover, it has an overhead of huge delay, area and power although it is more compact than [16,20] in terms of number of gates, quantum cost and garbage outputs.

4. Motivation Towards the New Concept of Reversible Online Testing

In this paper, the scope of the optimization of time complexity as well as eradication of garbage outputs and constant inputs has been addressed, which has motivated to introduce a completely new and an efficient technique of reversible online testing with the optimum time complexity, where the proposed technique works like the existing Dynamic Problem (DP) Algorithm [9].The reduction of time complexity minimizes the number of operations and hardware complexities. To implement the reversible online testable circuits, we proposed four reversible blocks, namely, Input Copying Block (ICB), Output

Copying Block (OCB), Parity Preserving Circuit Block and Myriad Feynman Gate Block (MFGB)in order to serve all reversible ESOP functions according to our proposed algorithm, where the ICB block generates the copies of input variables that will be needed in circuit for copying the same inputs and the OCB block copies the outputs to get the final required outputs and hence it degrades the number of gates and quantum cost. Thus, the number of gates, delay as well as area, power and the number of transistors are reduced. A large number of ancillary inputs and garbage outputs results in increasing the area, delay [21], power consumption and quantum cost [1, 7]. So, we are motivated to design our reversible online testable circuit in such a way that removes ancillary inputs [6] and garbage outputs [12, 22]. Parity preserving circuit block preserves parity bits and MFGB block performs the EXOR operation on the garbage outputs for checking online testability and thus, it prohibits garbage outputs from going through output lines. Memory elements store ancillary inputs and hence eliminate constant input lines. Omitting the garbage outputs and ancillary inputs along with introducing new technique for online testability, the proposed technique obtains a circuit with the optimum numbers of gates and transistors and reduced delay, area, and power consumption that makes the proposed technique a novel and an efficient technique in the literature so far.

5. Our Proposed Approach

This section introduces the proposed approach which can convert an ESOP function into an online testable circuit. In this section, all proposed reversible circuits have been described. In addition, Algorithm I is introduced to implement a reversible online testable circuit.

5.1 Technique for Constructing Reversible Online Testable Circuits with No Ancillary Inputs and Garbage Outputs

Reversible circuits of many ancillary inputs and garbage outputs are extremely difficult to realize [11-12, 22]. Reducing garbage outputs and constant inputs help to minimize the logical complexity and quantum costs of reversible circuits. Moreover, it becomes easier to design larger quantum circuits in a faster, compact and efficient way. In reversible circuits, a huge number of ancillary (constant) inputs and garbage outputs lead to increase the size (area) of the circuit as well as the gate level delays and power consumption. This sets the major objective of eliminating ancillary (constant) inputs and the number of the garbage outputs in the reversible online testable circuits [6, 12, 22].

In this paper, only a constant input is stored using a memory location [6, 11-12] which is supplied simultaneously to different blocks as an ancillary input according to the need of the circuits. Therefore, the number of ancillary input wires can be minimized using memory elements in the proposed reversible online testable circuit [6, 11-12]. As a result, the proposed reversible online testable circuit truncates ancillary inputs. Moreover, the required outputs only passes through the output wires and garbage outputs go through the proposed MFGB block and an EXOR operation is performed among those garbage outputs to ensure online testability checking. So, in output lines, no garbage comes out, i.e., the circuit is free from garbage outputs. Hence, the proposed design of the reversible online testable circuit discards ancillary inputs and overheads of garbage outputs which is first ever in the research of online testability till now.

5.2 Input Copying Block (ICB)

In this section, we discuss a reversible block, namely, Input Copying Block (ICB)with its block diagram as shown in Fig.3(a) and Fig.3(b), respectively. The whole circuit of this block is to be considered as a single reversible gate, although the block consists of *(ni-1)* numbers of *2*-input CNOT or Feynman gates, where $1 \leq i \leq n$ and n is the number of input lines for each value of i. Each input line of the ICB block is connected via a *2*-input CNOT gate and forms the final circuit which works as an input copying circuit. For example, an input of the given ESOP function is connected to the input line I_{11} of the ICB block and other inputs (I_{1i}) of the ICB block are initialized by $0(1 \leq i \leq n)$, where the outputsO_{1i} corresponding to the inputs I_{1i}($1\leq i \leq n$) produce the same outputs as the output of theI_{11} input line. *By* repeating the same procedure, we can obtain the other outputs from their corresponding inputs.

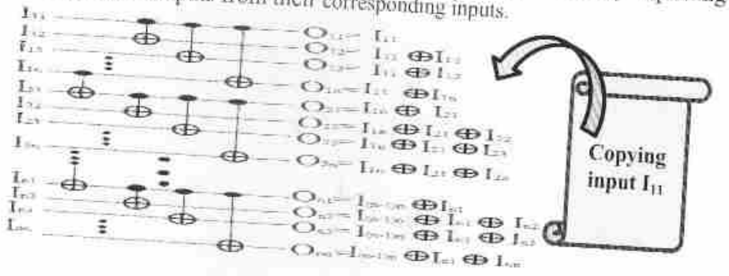


Fig. 3(a): Internal circuit of input copying block (ICB)

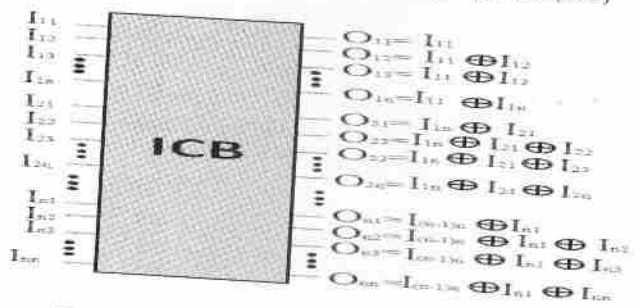


Fig. 3(b): Block diagram of input copying block (ICB)

Lemma 1: *ICB is a reversible block.*

■

Proof: We prove the above statement by the method of mathematical induction. The proposed reversible circuit ICB is considered as a single reversible gate. In order to prove the reversibility of the ICB block, we need to show that all the inputs have the one-to-one correspondence to its outputs, i.e., $I_{ij}\leftrightarrow O_{ij}$, where $1\leq i \leq n$, $1\leq j \leq n$ and n is the number of input lines for each value of I of the ICB block.
When $i=1$ and $1\leq j \leq n$, the input and output mapping will be $I_{1j}=(I_{11}, I_{12}, I_{13}, ..., I_{1n})$ and $O_{1j}=(I_{11}, I_{11}\oplus I_{12}, I_{11}\oplus I_{13}, ... I_{11}\oplus I_{1n})$, respectively, where all the inputs have the one-to-one correspondence to its outputs such as

$$I_{1j}\leftrightarrow O_{1j}.$$

So, the statement holds for base case $i=1$ and $1\leq j \leq n$. Assume that the statement also holds for $i=(n-1)$ and $1\leq j \leq n$.
So, when $i=(n-1)$ and $1\leq j \leq n$, the input and output mapping will be $I_{(n-1)j}=(I_{(n-1)1}, I_{(n-1)2}, I_{(n-1)3}, ..., I_{(n-1)n})$ and $O_{(n-1)j}=(I_{(n-2)n}\oplus I_{(n-1)1}, I_{(n-2)n}\oplus I_{(n-1)1}\oplus I_{(n-1)2}, I_{(n-2)n}\oplus I_{(n-1)1}\oplus I_{(n-1)3}, ..., I_{(n-2)n}\oplus I_{(n-1)1}\oplus I_{(n-1)n})$, respectively, where all the inputs have the one-to-one correspondence to its outputs such as

$$I_{(n-1)j}\leftrightarrow O_{(n-1)j}.$$

Therefore, when $i=n$ and $1\leq j \leq n$, the input and output mapping will be $I_{nn}=(I_{n1}, I_{n2}, I_{n3}, ... I_{nn})$ and $O_{nn}=(I_{(n-1)n}\oplus I_{n1}, I_{(n-1)n}\oplus I_{n1}\oplus I_{n2}, I_{(n-1)n}\oplus I_{n1}\oplus I_{n3}, ... I_{(n-1)n}\oplus I_{n1}\oplus I_{nn})$, respectively, where all the inputs have the one-to-one correspondence to its outputs such as

$$I_{nn}\leftrightarrow O_{nn}.$$

So, the statement also holds for $i=n$ and $1\leq j \leq n$.
Therefore, according to the method of mathematical induction, it can be stated that

$$I_{ij}\leftrightarrow O_{ij}, \text{ where } 1\leq i \leq n \text{ and } 1\leq j \leq n.$$

So, ICB is a reversible block and Lemma 1 is true

□

inputs $I_{i1}(1\leq i\leq n)$ produce the same outputs as the output of the I_{11} input line. By repeating the same procedure, we can obtain the other outputs from their corresponding inputs.

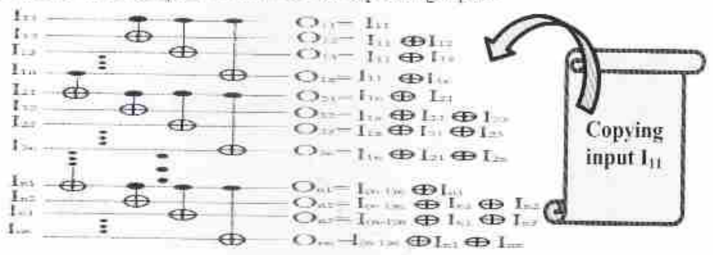


Fig. 3(a): Internal circuit of input copying block (ICB)

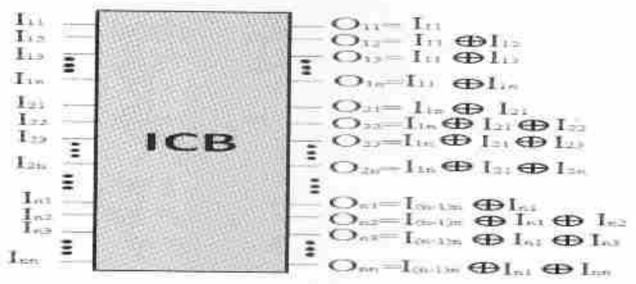


Fig. 3(b): Block diagram of input copying block (ICB)

Lemma 1: *ICB is a reversible block.*

■

Proof: We prove the above statement by the method of mathematical induction. The proposed reversible circuit ICB is considered as a single reversible gate. In order to prove the reversibility of the ICB block, we need to show that all the inputs have the one-to-one correspondence to its outputs, i.e., $I_{ij}\leftrightarrow O_{ij}$, where $1\leq i\leq n$, $1\leq j\leq n$ and n is the number of input lines for each value of I of the ICB block.

When $i=1$ and $1\leq j\leq n$, the input and output mapping will be $I_{1j}=(I_{11}, I_{12}, I_{13},..., I_{1n})$ and $O_{1j}=(I_{11}, I_{11}\oplus I_{12}, I_{11}\oplus I_{13},... I_{11}\oplus I_{1n})$, respectively, where all the inputs have the one-to-one correspondence to its outputs such as

$$I_{1j}\leftrightarrow O_{1j}.$$

So, the statement holds for base case $i=1$ and $1\leq j\leq n$. Assume that the statement also holds for $i=(n-1)$ and $1\leq j\leq n$.

So, when $i=(n-1)$ and $1\leq j\leq n$, the input and output mapping will be $I_{(n-1)j}=(I_{(n-1)1}, I_{(n-1)2}, I_{(n-1)3},..., I_{(n-1)n})$ and $O_{(n-1)j}=(I_{(n-2)n}\oplus I_{(n-1)1}, I_{(n-2)n}\oplus I_{(n-1)1}\oplus I_{(n-1)2}, I_{(n-2)n}\oplus I_{(n-1)1}\oplus I_{(n-1)3},..., I_{(n-2)n}\oplus I_{(n-1)1}\oplus I_{(n-1)n})$, respectively, where all the inputs have the one-to-one correspondence to its outputs such as

$$I_{(n-1)j}\leftrightarrow O_{(n-1)j}.$$

Therefore, when $i=n$ and $1\leq j\leq n$, the input and output mapping will be $I_{nn}=(I_{n1}, I_{n2}, I_{n3},... I_{nn})$ and $O_{nn}=(I_{(n-1)n}\oplus I_{n1}, I_{(n-1)n}\oplus I_{n1}\oplus I_{n2}, I_{(n-1)n}\oplus I_{n1}\oplus I_{n3},... I_{(n-1)n}\oplus I_{n1}\oplus I_{nn})$, respectively, where all the inputs have the one-to-one correspondence to its outputs such as

$$I_{nn}\leftrightarrow O_{nn}.$$

So, the statement also holds for $i=n$ and $1\leq j\leq n$.

Therefore, according to the method of mathematical induction, it can be stated that

$$I_{ij}\leftrightarrow O_{ij}, \text{ where } 1\leq i\leq n \text{ and } 1\leq j\leq n.$$

So, ICB is a reversible block and Lemma 1 is true □

5.3 Parity Preserving Block

This block is a circuit implemented by the given reversible ESOP function. It takes the inputs from the ICB block and produces the required outputs as well as garbage outputs, where the required outputs are copied by the OCB block and the garbage outputs are EXORed in the MFGB block with the other garbage outputs and the initial value of the MFGB block. Moreover, it checks whether the parity of all inputs of the given circuit and the parity of all outputs of the same circuit are equal or not.

5.4 Output Copying Block (OCB)

In this section, we discuss a reversible block for copying the final required outputs of the given ESOP function generated by the parity preserving circuit, namely, Output Copying Block (OCB) with its block diagram as shown in Fig. 4(a) and Fig. 4(b), respectively. The whole circuit of the OCB block is to be considered as a single reversible gate consisting of *(n-1)* numbers of 2-input CNOT or Feynman gates, where *n* is even number of input lines. In the OCB block, each input line is connected via a 2-input CNOT gate and forms the final circuit which works as an output copying circuit. Suppose, an output generated by the parity preserving block from an output of the given ESOP function is connected to the input line $I_1$of the OCB block and the input I_2 of OCB block is initialized by 0, where the outputs O_1 and O_2corresponding to the inputs I_1 and I_2 produce the same output. *By* repeating the same procedure, we can obtain the other outputs from their corresponding inputs.

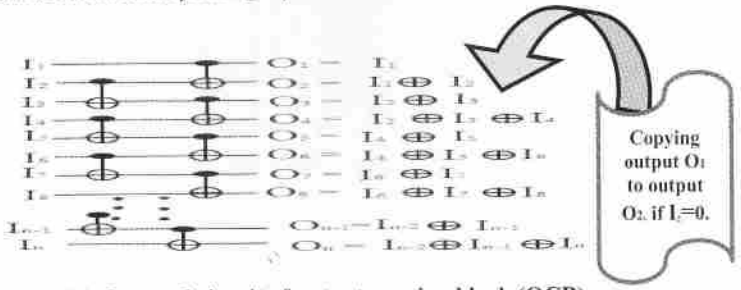


Fig. 4(a): Internal circuit of output copying block (OCB)

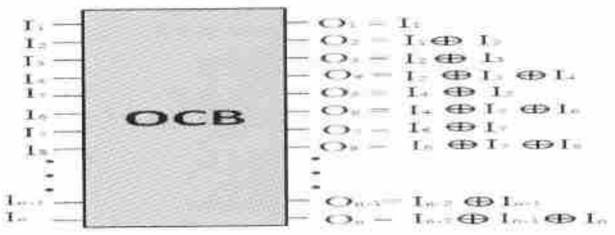


Fig. 4(b): Block diagram of output copying block (OCB)

Lemma 2: *OCB is a reversible block.*

■

Proof: We prove the above statement by the method of mathematical induction. The proposed reversible circuit OCB is considered as a single reversible gate. It will be reversible *iff* all the inputs have the one-to-one correspondence to its outputs, i.e., $I_n \leftrightarrow O_n$,where *n* is even numbers of input lines of the OCB block. When *n=2*, the input and output mapping will be $I_2=(I_1, I_2)$ and $O_2=(I_1, I_1 \oplus I_2)$, where all the inputs have the one-to-one correspondence to its outputs as follows:

$$I_2 \leftrightarrow O_2.$$

So, the statement holds for base case *n=2*.
Assume that the statement also holds for *n=(n-2)*.

So, when $n=(n-2)$,the input and output mapping will be $I_{(n-2)}=(I_1, I_2 ... I_{(n-3)}, I_{(n-2)})$ and $O_{(n-2)}=(I_1, I_1 \oplus I_2 ... I_{(n-4)} \oplus I_{(n-3)}, I_{(n-4)} \oplus I_{(n-3)} \oplus I_{(n-2)})$, where all the inputs have the one-to-one correspondence to its outputs as follows:

$$I_{(n-2)} \leftrightarrow O_{(n-2)}.$$

Therefore, when $n=n$, the input and output mapping will be
$I_n=(I_1, I_2 ... I_{n-1}, I_n)$ and $O_n=(I_1, I_1 \oplus I_2 ... I_{(n-2)} \oplus I_{(n-1)}, I_{(n-2)} \oplus I_{(n-1)} \oplus I_n)$, where all the inputs have the one-to-one correspondence to its outputs as follows:

$$I_n \leftrightarrow O_n.$$

So, OCB is a reversible block and Lemma 2 is true.

5.5 Myriad Feynman Gate Block (MFGB)

In this section, we propose another necessary reversible block, namely, Myriad Feynman Gate Block (MFGB) as shown in Fig. 5. The whole circuit of the MFGB block is to be considered as a single reversible gate consisting of $(n-1)$ numbers of 2-input CNOT or Feynman gates, wherenis the number of input lines and each input of the MFGB block is EXORed with the n^{th} input-output line of the MFGB block via a 2-input CNOT gate. For example, an output line of the OCB block is connected to an input line $I_1$of the MFGB block. Similarly, the other inputs of the MFGB block are produced, where the n^{th} output line (O_n) of the MFGB block generates $I_1 \oplus I_2 \oplus I_3 \oplus ... \oplus I_{n-1} \oplus I_n$.

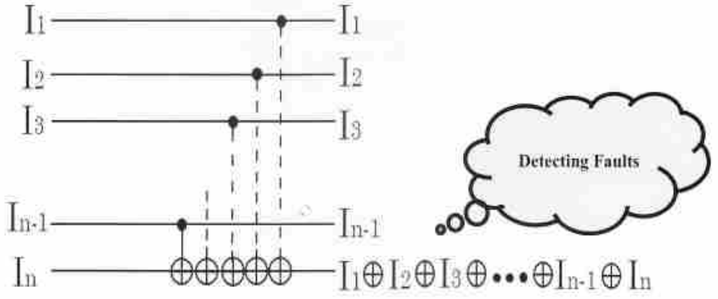


Fig. 5: Myriad feynman gate block (MFGB)

Lemma 3: *MFGB is a reversible block.*

■

Proof: We prove the above statement by the method of mathematical induction. The proposed reversible circuit MFGB is considered as a single reversible gate. When $n=2$, the input and output mapping will be as follows:

$$I_2=(I_1, I_2) \text{ and } O_2=(I_1, I_1 \oplus I_2),$$

where all the inputs have the one-to-one correspondence to its outputs as shown below:

$$I_2 \leftrightarrow O_2.$$

So, the statement holds for base case $n=2$.
Assume that the statement also holds for $n=(n-1)$.
So, when $n=(n-1)$,the input and output mapping will be
$I_{(n-1)}=(I_1,I_2,I_3,...,I_{(n-1)})$ and $O_{(n-1)}=(I_1,I_2,...I_{(n-1)},I_1 \oplus I_2 \oplus I_3 \oplus ... \oplus I_{n-1})$,
where all the inputs have the one-to-one correspondence to its outputs as follows:

$$I_{(n-1)} \leftrightarrow O_{(n-1)}.$$

Therefore, when $n=n$, the input and output mapping will be
$I_n=(I_1,I_2,I_3,...,I_n)$ and $O_n=(I_1,I_2,I_3,...I_n,I_1 \oplus I_2 \oplus I_3 \oplus ... \oplus I_{n-1} \oplus I_n)$, where all the inputs have the one-to-one

correspondence to its outputs as follows:

$$I_n \leftrightarrow O_n.$$

So, MFGB is a reversible block and Lemma 3 is true.

5.6 Proposed Online Testable Reversible Circuit

In this section, we design the proposed testable circuit by combining the four blocks, namely, ICB, OCB, Parity Preserving Circuit and MFGB. Firstly, the recurrence of each input variable of an ESOP expression is checked as the recurrence of such input variables is essential for ICB block to generate the number of copies of the recurrent input variable. Suppose, $\alpha_{i1}, \alpha_{i2}, \alpha_{i3}, ..., \alpha_{in}$ be the input variables of a reversible ESOP function, where $S\alpha_{i1}, S\alpha_{i2}, S\alpha_{i3}, ..., S\alpha_{in}$ are the numbers of copiesof$\alpha_{i1}, \alpha_{i2}, \alpha_{i3}, ..., \alpha_{in}$, respectively. Now, $S\alpha_{i1}, S\alpha_{i2}, S\alpha_{i3}, ..., S\alpha_{in}$ copies of input variables are EXORed with the initial value (P) of MFGB block. Additionally,$S\alpha_{i1}, S\alpha_{i2}, S\alpha_{i3}, ..., S\alpha_{in}$ copies of input variables pass through the MFGB block to the parity preserving reversible block to act as its inputs. Therefore, the total number of outputs generate from the parity preserving reversible block is $(S\alpha_{i1}+S\alpha_{i2}+S\alpha_{i3}+ ...+S\alpha_{m})$ as reversible circuit generates the same number of outputs as the number of inputs. If the garbage outputs or the unused outputs are Gi, then the total number of required outputs is $(S\alpha_{i1}+S\alpha_{i2}+S\alpha_{i3}+ ...+S\alpha_{in})-Gi$. It is important to note that the required number of outputs will be used as the final outputs. Additionally, it is also copied by the OCB block and the copied outputs along with the garbage outputs will be EXORed with the initial value (P) of the MFGB block to detect the error of the proposed on-line testable circuit. It is clear from the above discussion that all the garbage outputs from different blocks of the proposed online testable circuit are EXORed in the MFGB block which produces an output for detecting the fault of the proposed online testable circuit. Thus, the proposed circuit is garbage free. Moreover, a single line for the ancillary input is used in the proposed online testable circuit, where the value of the ancillary input is copied into a memory block, which is further supplied to the necessary locations of the proposed circuit. So, the proposed circuit can be treated as free from ancillary inputs [6].The proposed online testable circuit is shown in Fig. 6.Table 1 shows the comparative result analysis among the proposed method and the existing ones [16, 18, 20].

Algorithm 1: Algorithm for the Proposed Reversible Online Testable Approach.

Let the input variables be $\alpha_1, \alpha_2, \alpha_3 ... \alpha_n$, the ESOP expression be f, counter function be $c(fx_i)$ and storing variable be $S\alpha_n$ where $1 \le i \le n$, $1 \le j \le n$ and n is a natural number.

INPUT: Circuit Inputs I_{ij}

OUTPUT: Circuit Outputs O_{ij} and Garbage Outputs G_i

```
1. Begin
  2. Construct the partial product functions as follows:
    $f=\vee_{i=1}^{i=n}(\alpha 1 ... \alpha i..\alpha n) \oplus$
    $\vee_{i=1}^{i=n}(\alpha 1, ... \alpha i, ... \alpha n)(\alpha 1, ... \alpha i, ... \alpha n) \oplus \vee_{i=1}^{i=n}(\alpha 1, ... \alpha i, ... \alpha n)(\alpha 1, ... \alpha i, ... \alpha n)(\alpha 1, ... \alpha i, ... \alpha n)$
    $\oplus$
    $... \oplus \vee_{i=1}^{i=n}(\alpha 1, ... \alpha i, ... \alpha n)(\alpha 1, ... \alpha i, ... \alpha n) ... (\alpha 1, ... \alpha i, ... \alpha n).$
    Let $fx_1 = \vee_{i=1}^{i=n}(\alpha 1 ... \alpha i..\alpha n)$
    $fx_2 = \vee_{i=1}^{i=n}(\alpha 1, ... \alpha i, ..\alpha n)(\alpha 1, ... \alpha i, ... \alpha n)$
    $\vdots$
    $fx_n = \vee_{i=1}^{i=n}(\alpha 1, ... \alpha i, ... \alpha n)(\alpha 1, ... \alpha i, ... \alpha n) ... (\alpha 1, ... \alpha i, ... \alpha n);$
    where$fx_1, fx_2, ..., fx_n$ are the partial product functions.
```

3. Generate the EXORs of the partial product functions as follows:

$$f = fx_1 \oplus fx_2 \oplus fx_3 \oplus fx_4 \oplus \ldots fx_n.$$

where the numbers of the recurrences of the input variables in ESOP expression are

$$S\alpha_1 \leftarrow c_{\alpha 1}\,(fx_1, fx_2, fx_3, fx_4 \ldots fx_n)$$
$$S\alpha_2 \leftarrow c_{\alpha 2}\,(fx_1, fx_2, fx_3, fx_4 \ldots fx_n)$$
$$\vdots$$
$$S\alpha_n \leftarrow c_{\alpha n}\,(fx_1, fx_2, fx_3, fx_4, \ldots fx_n)$$

4. Insert the input variables of the ESOP expression into the inputs of the ICB block

$$I_{11} \leftarrow \alpha_1, I_{12} \leftarrow 0 \ldots I_{1n} \leftarrow 0, \text{ where } 1 \le n \le S\alpha_1$$
$$I_{21} \leftarrow \alpha_2, I_{22} \leftarrow 0 \ldots I_{2n} \leftarrow 0, \text{ where } 1 \le n \le S\alpha_2$$
$$\vdots$$
$$I_{n1} \leftarrow \alpha_n, I_{n2} \leftarrow 0 \ldots I_{nn} \leftarrow 0, \text{ where } 1 \le n \le S\alpha_n.$$

5. Do the EXOR operations of the initial value (P) of the MFGB block with the products of the input variables and Numbers of the recurrences of input variables of the ESOP expression as follows:

$$X = P \oplus \alpha_1.S\alpha_1 \oplus \alpha_2.S\alpha_2 \oplus \alpha_3.S\alpha_3 \oplus \ldots \oplus \alpha_n.S\alpha_n.$$

6. Generate the outputs from the Parity Preserving circuit as shown below

$$O_{ij} \rightarrow (\beta_1.S\alpha_1 + \beta_2.S\alpha_2 + \beta_3.S\alpha_3 + \ldots + \beta_n.S\alpha_n) \text{ as } I_{ij} \leftrightarrow O_{ij}.$$

where β_i is the output of the Parity Preserving Block.

7. Generate the total required outputs excluding the garbage outputs from the Parity Preserving Circuit as shown below:

$$O_{ij} - Gi = (\beta_1.S\alpha_1 + \beta_2.S\alpha_2 + \beta_3.S\alpha_3 + \ldots + \beta_n.S\alpha_n) - Gi.$$

8. Copy the total required outputs generated in Step 7 by the OCB block as follows:

$$I_1 \leftarrow \beta_1, I_2 \leftarrow 0, I_3 \leftarrow \beta_2 I_4 \leftarrow 0 \ldots I_{k-1} \leftarrow \beta_n, I_k \leftarrow 0$$
$$O_1 \rightarrow \beta_1, O_2 \rightarrow \beta_1, O_3 \rightarrow \beta_2\, O_4 \rightarrow \beta_2 \ldots O_{k-1} \rightarrow \beta_n, O_k \rightarrow \beta_n$$

where k is even.

9. Generate the outputs including the garbage outputs from the Parity Preserving Block and insert those outputs into the MFGB block.

10. Generate the final outputs from the OCB block.

11. Generate the parity preserving value from the MFGB block. If the value is 1, the circuit is faulty, otherwise the circuit is fault free.

12. **End**

Example: Consider the following reversible ESOP function: $f(a,b,c)=abc\oplus c'$. Fig. 7 (a) shows the block diagram of the proposed reversible online testable circuit of the given reversible ESOP function, whereas Fig. 7 (b) shows the detailed implementations of the individual blocks of Fig. 7 (a) using **Algorithm 1**.

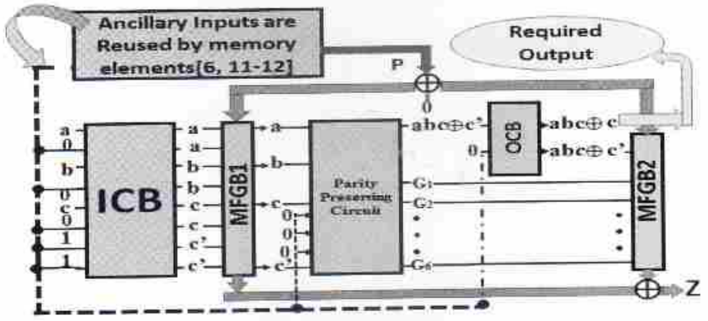

Ancillary Inputs are Reused by memory elements[6, 11-12]
Required Output
P
ICB
MFGB1
Parity Preserving Circuit
OCB
MFGB2
abc⊕c'
Z

Fig. 7(a): Block diagram of the proposed reversible online testable circuit of the given ESOP function $f(a,b,c)=abc\oplus c'$

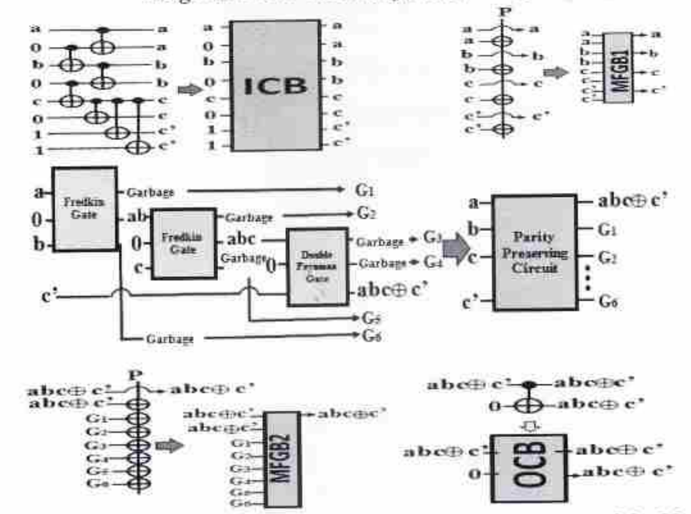


Fig. 7(b): Detailed implementations of the individual blocks of the block diagram shown in Fig. 7(a)

Theorem1: The proposed Reversible Online Testable Algorithm requires $O(log_2 n)$ complexity of time, where n is the number of inputs.

■

Proof: Assume that n be the number of inputs. The proposed algorithm is defined as follows:

$$T(n) = 2.T(\lfloor\sqrt{n}\rfloor) + log_2 4;$$

where a constant term $log_2 4$ is used to indicate the number of discarded constant input lines. Then, we can simplify the recurrence with a change of variables. For the recurrence, let $m = log_2 4$. The value of m yields

$$T(2^m) = 2.T(2^{m/2}) + m.$$

We can now rename the recurrence with $S(m) = T(2^m)$to produce a new recurrence which is as follows:

$$S(m) = 2.S(m/2) + m.$$

Indeed, the new recurrence has the solution as mentioned below:

$$S(m) = O(m.\log m)$$

So, $T(n) = T(2^m) = S(m) = O(m.\log m)$

$$= O(\log_2 4.\log(\log_2 4))$$
$$= O(2.\log 2).$$

Now, assume that, $n = 100$*, and replace* $n=100$*by* $n=\log^{-1}2$.

Hence, we get $2 = \log n$ *since* $\log 100 = 2$.

So, the resulting recurrence is:

$$T(n) = O(2.\log 2)$$
$$= O(\log n.\log 2)$$
$$= O(\log_2 n).$$

$$[Since\ (\log_k n).(\log_k 2) = \log_2 n]$$

Finally, the complexity of the proposed reversible online testable algorithm is obtained which is $O(\log_2 n)$.
□

Theorem 2: The proposed circuit is Online testable that can detect any single bit fault while the circuit is operating. ■

Proof: Let I_i be the set of inputs and O_i be the set of outputs that are implemented by a reversible function. Let the input lines be $I_1, I_2 \ldots I_n$ and output lines be $O_1, O_2 \ldots O_n$. The parity preserving property states that

$$I_1 \oplus I_2 \oplus I_3 \oplus \ldots \oplus I_n = O_1 \oplus O_2 \oplus O_3 \oplus \ldots \oplus O_n$$

If the online testable circuit is faulty, the final outputs are inverted and denoted as O_i'.

The required outputs are copied by the OCB. If G_i is the set of garbage outputs, then the total required outputs are

$$O_i - G_i.$$

In the Parity Preserving Band, the total outputs are as follows:

$$(O_i - G_i) + G_i = O_i$$

Therefore, the final output (*Z*) of the Parity Preserving Band is

$$= I_i \oplus ((O_i - G_i) + G_i) = I_i \oplus O_i = 0$$

If the final output (*Z*) *is* 0, then there is no fault in the circuit.

If any fault is generated within the Parity Preserving Band, then total outputs are

$$(O_i'-G_i)+G_i=O_i'$$

Therefore, the final output (Z) of the Parity Preserving Band is

$$=I_i\oplus((O_i'-G_i)+G_i)=Ii\oplus O_i'=I$$

Thus, if any fault is occurred, the final output (Z) is 1 during the operation of the reversible circuit.

Therefore, Theorem 2 is true and it is proved.

□

5.7 Comparison of the Proposed Online Testable Technique with the Existing Ones

Table 1 shows the comparative result analysis among the proposed method and the existing ones [16, 18, 20]. From this table, we can see that our method is much better than the existing methods.

Table 1: Comparison between the proposed and existing methods of reversible online testability in terms of time complexity

Method	Time
Proposed Algorithm	$O(log_2 n)$
Existing [18]	$O(n)$
Existing [20]	$O(n^2)$
Existing [16]	$O(n^3)$

6. Simulation Results and Comparison

To test the efficiency of proposed online testable reversible circuit, several benchmark functions [8]have been used on a computer which has the Intel(R) Core(TM) 2 Duo CPU T6570, 2.10 GHz clock speed and 1.92 GB of RAM. The simulation is done using Microwind DSCH 3.5 [14], STM Cell Library [13] and Algorithm 1. Existing approach [16] has no synthesized technique. Two of the existing approaches [16] and [20] present a cascading of specific reversible gates to construct the reversible online testable circuit. However, we have used the traditional logic gates such as AND, OR, NAND and NOR to implement the benchmark functions and generated a technique to replace these traditional gates with the specific reversible gates.

Online testability on ESOP circuit is proposed in [18], where each CNOT gate is replaced by traditional gates. In our proposed approach, each benchmark function with the ESOP form is converted into a testable circuit using traditional Boolean gates.

Table 2, Table 3, Table 4 and Table 5 show that our approach minimizes area, power, delay, number of transistors, quantum cost, number of gates and produce the final circuit with no garbage outputs and ancillary inputs. Our approachshows its efficiency overthe best known existing approaches [16,18,20] with respect to all performance matrices.

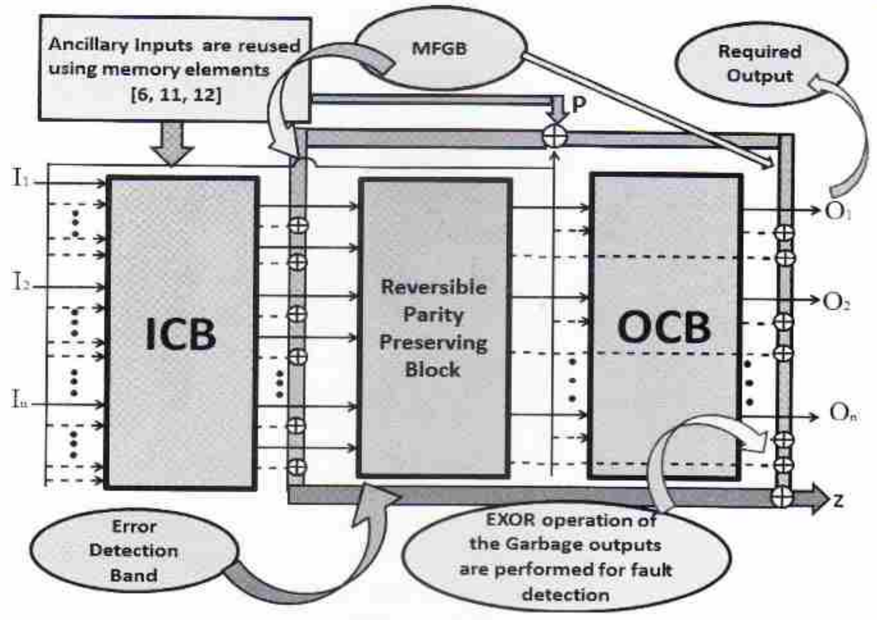

Ancillary Inputs are reused using memory elements [6, 11, 12]
MFGB
Required Output
P
I_1
I_2
I_n
ICB
Reversible Parity Preserving Block
OCB
O_1
O_2
O_n
Z
Error Detection Band
EXOR operation of the Garbage outputs are performed for fault detection

Fig. 6: Proposed online testable reversible circuit

Table 2: Comparison of area and power of the proposed approach with the existing approaches using benchmark functions

Benchmark Function	Area (in μm^{2})				Power (in mW)			
.	Existing [16]	Existing [20]	Existing [18]	Proposed	Existing [16]	Existing [20]	Existing [18]	Proposed
xor5	324.096	100.02	33.6	32	51.2	20.05	4.2	4.1
majority	1377.40	500.78	200.584	43.704	217.6	78.55	35.8	7.3
con1	2916.86	801.31	392.952	226.6	460.8	250.92	74.2	33.3
T481	3159.93	2206.20	443.904	244.192	499.2	260.75	78.2	35.2
rd53	3000	2018.45	266.976	248.472	444	315.49	47.4	34.9
c17	1539.45	945.29	189.312	130.632	215.7	125.32	32.6	18.5
ex2	1701.504	730.82	215.512	134.896	218.8	140.68	38	19.2
rd32	1782.528	1032.89	221.696	148.224	281.6	165.75	38.4	21.2
f2	3159.936	1945.73	438	270.32	499.2	250.50	79.8	38.8
cm82a	2592.768	1309.34	265.92	218.08	409.6	200.48	46.6	30.6

Table 3: Comparison of delay and number of transistors of the proposed approach with the existing approaches using benchmark functions

Benchmark Function	Delay *(in ns)*				Number of Transistors			
	Existing [16]	Existing [20]	Existing [18]	Proposed	Existing [16]	Existing [20]	Existing [18]	Proposed
xor5	20.48	10.65	2.56	1.28	1024	265	84	82
majority	40.96	20.391	12.64	2.88	4352	2024	716	146
con1	46.08	28.15	23.04	6.24	9216	5128	1324	666
T481	125.44	50.63	28.48	6.88	9984	6135	1564	704
rd53	51.2	35.2	16.64	6.4	8880	6012	948	698
c17	30.72	21.6	11.68	3.52	4314	3156	652	370
ex2	56.32	30.72	13.44	3.84	4376	2725	760	384
rd32	25.6	16.88	13.44	4	5632	3085	768	424
f2	30.72	18.56	26.88	6.72	9984	7102	1596	776
cm82a	51.2	35.60	16.48	5.6	8192	5132	892	612

Table 4: Comparison of quantum cost and garbage output of the proposed approach with the existing approaches using benchmark functions

Benchmark Function	Quantum Cost				Garbage Output			
	Existing [16]	Existing [20]	Existing [18]	Proposed	Existing [16]	Existing [20]	Existing [18]	Proposed
xor5	232	36	20	20	144	8	4	0
majority	787	224	154	108	177	18	5	0
T481	2262	556	415	286	468	108	16	0
c17	486	190	169	144	109	23	5	0
ex2	1733	267	171	152	387	23	5	0
f2	1948	533	306	308	434	44	4	0
cm82a	2163	370	290	235	482	45	5	0

Table 5: Comparison of the number of gates of the proposed approach with the existing approaches using benchmark functions

Benchmark Function	Number of Gates			
	Existing [16]	Existing [20]	Existing [18]	Proposed
xor5	48	22	13	8
majority	204	52	22	12
T481	468	96	63	25
c17	228	44	22	13
ex2	252	64	29	21
f2	468	102	37	23
cm82a	384	78	42	26

7. Conclusion

This paper has presented the efficient design synthesis of online testable reversible circuits with the optimum time complexity. In addition, we have proposed three reversible blocks, namely, Input Copying Block (ICB), Output Copying Block (OCB) and Myriad Feynman Gate Block (MFGB). We have shown the efficiency of proposed designs over existing approaches using benchmark functions [8]. We have also proved the correctness of proposed designs by several lemmas and theorems. The unique feature of the presented online testable reversible circuit is that it requires no garbage outputs and ancillary inputs. In addition, we have presented an algorithm to construct the proposed reversible online testable circuit using the proposed circuit blocks. It has also been shown that the proposed circuit requires the optimum area, power, delay, number of transistors, gates, quantum cost and delay. As the testability of reversible circuit becomes a major issue as well as its applications in various areas such as FPGA, BCD adders, dividers, ALU [23]etc., the proposed technique may help to achieve the cost efficient reversible circuit with online testability.

References

[1] M. Nielson and I. Chuang. "Quantum computation and quantum information", Cambridge University Press, 2000.
[2] R. Landauer, "Irreversibility and heat generation in the computional process", IBM Journal of Research and Development, 5, pp.183 -191, 1961.
[3] L. D. Paulson, "Reversible computing may improve mobile performance", *Computer*, vol. 37, no. 3, p. 21, Mar. 2004.
[4] M. Perkowski and P. Kerntopf, 2001, "Revesible Logic", Invited tutorial, proc. EURO-MICRO ,Sept 2001, Warsaw, poland.
[5] C. H. Bennett, "Logical reversibility of Computation", IBM J. Res. Dev., 17:525-532, 1973.
[6] Y. Takahashi and N. Kunihiro, "A linear-size quantum circuit for additionwith no ancillary qubits", Quantum Information and Computation, vol. 5,no. 6, pp. 440–448, 2005.
[7] J. Stolze and D. Suter, "Quantum computing: a short course from theory to experiment", Wiley-VCH Gmb H& Co, KGaA, Weinheim (2004).
[8] D. Maslov (2012) Reversible logic synthesis benchmarks page. http://www.cs.uvic.ca/dmaslov/
[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms", Cambridge, MA: MIT Press, 2001.
[10] Barenco, Adriano, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter. "Elementary gates for quantum computation", Physical Review, *A*52, no. 5 (1995): 3457.
[11] Cuccaro, S. A., Draper, T. G., Kutin, S. A., and Moulton, D. P., "A new quantum ripple-carry addition circuit", *http://arXiv.org/quant-ph/0410184,* Oct2004.
[12] Y. Takahashi, S. Tani, and N. Kunihiro, "Quantum addition circuits and unbounded fan-out", *http://arxiv.org/abs/0910.2530*, Oct 2009.
[13] L. Dadda, 'Multioperand Parallel Decimal Adder: A Mixed Binary and BCD Approach", IEEE Transactions on Computers, vol. 56, no. 10, pp. 1320-1328, Oct. 2007.
[14] Microwind DSCH – Schematic Editor and Digital Simulator, http://www.microwind.net/dsch.ph.
[15] K. N. Patel, J. P. Hayes and I. L. Markov, "Fault testing for reversible circuits", In the Proceedings of the 21st IEEE VLSI Test Symposium, pp. 410–416, April 2003.
[16] D. P. Vasudevan, P. K. Lala, D. Jia and J. P. Parkerson, "Reversible logic design with online testability", IEEE Trans InstrumMeas, vol. 55, no. 2, pp. 406–414, 2006.
[17] K. Fazel, M. Thornton, and J. E. Rice, ""ESOP-based Toffoli gate cascade generation", In: Proceedings of the IEEE Pacific Rim conference on communications, computers and signal processing (PACRIM). Victoria, pp 206–209, 2007.

[18] N. M. Nayeem and J. E. Rice, "Online Testable Approaches in Reversible Logic", Journal of Electronic Testing, vol. 29, no. 6, pp. 763-778, 2013.

[19] L .Wang, C. Wu, and X. Wen, "VLSI test principles and architectures: design for testability", Morgan Kaufmann.

[20] S. N. Mahammad and K.Veezhinathan, "Constructing online testable circuits using reversible logic", IEEE Trans InstrumMeas, vol. 59, no. 1, pp. 101–109, 2010.

[21] P. Mroszczyk and P. Dudek, "Tunable CMOS delay gate with improved matching properties", IEEE Trans Circuits and Systems, vol. 61, issue 9, 2014.

[22] D. Maslov and G. W. Dueck, "Reversible cascades with minimal garbage", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, pp. 1497–1509, November 2004.

[23] Morrison, Matthew, and N. Ranganathan. "Design of a reversible ALU based on novel programmable reversible logic gate structures", IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2011.

[24] D. K. Kole, H. Rahaman and D. K. Das, "Synthesis of online testable reversible circuit", In: Proceedings of 13th IEEE international symposium on design and diagnostics of electronic circuits and systems (DDECS). Vienna, pp. 277–280, 2010.

Author's Biography

Professor Dr. Hafiz Md. Hasan Babu is currently working as a Professor in the Department of Computer Science & Engineering, University of Dhaka, Bangladesh. He is also the former Chairman of the same department. Professor Dr. Hasan Babu is a Member (Part-Time) of Bangladesh Accreditation Council of Government of the People's Republic of Bangladesh. He is a former Professor and the Founding Chairman of the Department of Robotics and Mechatronics Engineering, University of Dhaka, Bangladesh. He is also the Former Pro-Vice-Chancellor of National University, Gazipur, Bangladesh. Professor Dr. Hasan Babu obtained his Ph.D. in Electronics and Computer Science from Japan under the Japanese Government Scholarship and received his M.Sc. in Computer Science and Engineering from Czech Republic under the Czech Government Scholarship.

Professor Dr. Hasan Babu is currently an Associate Editor of the famous research journal of the United Kingdom "IET Computers and Digital Techniques". He was awarded the Bangladesh Academy of Sciences Dr. M.O. Ghani Memorial Gold Medal Award in 2017 for his excellent research work to the progress of Physical Sciences in Bangladesh.

Professor Dr. Hafiz Md. Hasan Babu published more than a hundred research papers. Three of his research papers got the best paper award. Professor Dr. Hasan Babu was also a member of Prime Minister's ICT Task Force. At present, he is the president of Bangladesh Computer Society and also the President of International Internet Society, Bangladesh Chapter. He has been awarded the UGC Gold Medal Award in 2017 in Mathematics, Statistics and Computer Science category for his research work on quantum multiplier-accumulator device. He has published two text books: 1) "Quantum Computing: A Pathway to Quantum Logic Design" by IOP Publishing, United Kingdom and 2) "Reversible and DNA Computing" by Wiley Publishers, United Kingdom.

BCS Journal of Computer and Information Technology
Vol. 02, 2020, ISSN: 2664-4592 (Print), 2664-4606 (Online)

Distributed Denial of Service Attack Detection in Cloud Computing Using Machine Learning

Md. Mehedi Hasan and Md. Saiful Islam
Institute of Information and Communication Technology
Bangladesh University of Engineering and Technology
Dhaka-1205, Bangladesh
E-mail (s): mehedimec@gmail.com, mdsaifulislam@iict.buet.ac.bd

Abstract: Distributed Denial of Service (DDoS) is one of the most frequent attacks in cloud that cause significant damage, affect the performance and continue to be the predominant security challenge that compromise the availability of the network. Researchers are constantly paying attention to DDoS attack detection to reach an impervious and guaranteed safe cloud computing. Various machine learning approaches have been used recently to detect DDoS attack. But over the past decade, research on DDoS attack detection has focused on a few classes of these attacks. To address this problem, this research presents a machine learning (ML) based DDoS detection system considering most types of modern DDoS attack in cloud computing environment. For this, a new dataset is generated in the lab environment. This research incorporates five popular machine learning algorithms: Decision Tree, Random Forest, Naïve Bayes, Logistic Regression and Stochastic Gradient Descent. Our experimental results show that with Random Forest algorithm, the DDoS attack detection rate is above 99% with high precision and recall. Moreover, our proposed method shows better performance compared to several machine learning based state-of-the-art methods presented in the literature.

Keywords: DDoS, Cloud Computing, Machine Learning, Decision Tree, Random Forest, Naïve Bayes, Logistic Regression, Stochastic Gradient Descent.

1. Introduction

Cloud computing is a ground-breaking idea which has changed information and communication technology by supplying computing resources, storage, infrastructure etc. as services over the internet. This technology delivers on demand resources according to the requirements of the system which is scalable and relatively less expensive. It replaces the requirement to invest for IT infrastructure and network for the organizations. But, security is a major concern in this cutting edge technology since a lot of systems are needed to keep safe in the cloud and more nodes are interconnected there are many possible entry points, and many systems to patch. Accordingly, ensuring security in cloud computing is much more critical than the traditional systems. One of the major threats in cloud computing is Denial of Service (DOS) attack. It is such an attack to the computing resources which degrades the service or makes it unavailable to the legitimate users. A more advanced type of DoS attack is Distributed Denial of Service (DDoS) attack, where multiple sources simultaneously launch an attack to the target system. As a result, the server load increases which results in system inaccessibility with in a very short time.

The absolute prevention of DDoS attacks in cloud quite impossible [1]. Therefore, detecting DDoS attacks is a significant initiative towards cloud security. In a DDoS attack in cloud, an attacker attempts to flood a target system by sending a huge amount of traffic from multiple virtual machines. Hence, the detection systems become unable to detect DDoS attacks effectively. Moreover, the distributed character of the DDoS attacks make it more challenging to identify. So, the mitigation of DDoS attacks remains very hard as it penetrates the security device and specially for cloud computing, it causes loss in terms of monetary and reputation for a company.

The DDoS attacks can be categorised based on network and application layer protocol. The network layer or transport layer attacks are generally launched by using the TCP, UDP and ICMP protocols, whereas the application layer attacks are basically application dependent like for webserver HTTP traffic is used, for DNS server DNS protocol is used etc.

So far researchers have been used many techniques for detecting DDoS attack. Among those, signature based and anomaly based detection are two of the most common approaches of defense mechanisms. In signature detection, a database of signatures of the previous attacks is maintained and attacks are detected by comparing the signature of an incoming attack with its signature database. Then it applies the defense mechanism for the detected attack. So, it is obvious that any new type of attack detection is almost impossible with this method. On the contrary, anomaly based DDoS detection techniques use a pre-defined threshold value in order to detect a DDoS attack and later differentiate the pattern of the attack with that threshold. This approach results too high false positives in the literature for DDoS attack detection. So, recent trend is to use ML techniques which are more accurate and fast in detecting DDoS attack.

Unfortunately, over the past decade, researchers have focused only some classes of DDoS attacks for detection [2]. To the best of our knowledge, datasets which include modern and most types of DDoS attack in cloud is not available. In our research, we have generated a new dataset that contains five types of DDoS attack including TCP SYN flood, UDP flood, DNS flood, ICMP flood and HTTP flood along with the legitimate traffic. We have presented a two stage feature selection method from the generated dataset by utilising Recursive Feature Elimination with Cross Validation and with our proposed feature selection algorithm. We applied machine learning algorithm for detection and multiclass classification of attacks based on the collected network traffic features of the dataset as well as with the extracted features.

Our contributions in this work are as follows:

- A novel dataset has been created which contains modern and diverse types of attack in cloud computing environment. There are 25 features and 06 classes in the dataset, where five most critical type of DDoS attack in the network, transport and application layers are considered along with the legitimate traffic.
- The correlation between five classification algorithms are explored by using an approach to automatic feature selection and cross validation for sorting a list of important features and measuring the accuracy of each classification algorithm.
- A new algorithm has been developed for automated feature selection from the dataset. Later machine learning algorithm is applied on the selected feature to detect DDoS attack, which .shows high accuracy.
- Performance comparison between several machine learning methods in addition with some other state of the art methods used in the literature to validate the obtained results.

The rest of this paper is organized as follows: Section 2 presents a summary of the previous related studies. In Section 3, our experimental setup, attack generation and dataset collection process are described. Section 4 discusses the process of feature engineering and Machine Learning Algorithms (MLA) selection of our work. Section 5 presents the process of the evaluation metrics calculation to assess performance. In section 6, our achieved results are presented and we conclude our work in section 7.

2. Related Work

The current increase of DDoS attacks has allured a significant attention to the researchers. Different methods have been created for preventing these attacks on the cloud like unseen servers, challenge answer, precautionary access and to limit the resources. Other approaches that are adopted to defend DDoS attacks are puzzle based techniques like CAPTCHA, that provides a simple process to mitigate attack. But it is proven ineffective by the recent works [3]. Another technique that is used consistently is to monitor the IP address of the incoming packets for DDoS attacks detection [4]. In [5], a digital signature that uses meta-heuristic methods was considered in order to investigate network flow for anomaly detection. This model improved the DDoS attack detection accuracy but shows poor result for general DoS attacks detection. Again, IP trace back defence system which consists of packet identification, reassembling process and source detection based on entropy is used in [6]. Another study examined the potentiality of firewalls to defend DDoS attacks in the cloud [7]. This factual study figured it out that neither software nor hardware based firewalls are sufficient to mitigate DDoS attack.

Various machine learning and data mining techniques have recently been used for the detection and prevention of DDoS attacks. In [8], a performance analysis of some unsupervised and supervised Machine Learning Algorithms (MLA) for DDoS attacks detection is discussed. In this study, authors collect information from server, proposes a defense mechanism using machine learning. The method is applied near to the attacker's location in the cloud. Moreover, in order to increase the performance of classifier for attack detection, semi-supervised machine learning models are applied in [9].

To detect reflection DDoS attacks, authors in [10] discussed an approach which is not dependent on protocol. They have determined five significant features for the detection of Distributed Reflection Denial-of Service (DRDoS) for protocol-independent approach. Another machine learning based DDoS detection technique is presented in [11]. This work considers eight features from the CAIDA'07 dataset and applied naive Bayes algorithm. An anomaly based DDoS detection method utilizing network traffic analysis is suggested in [12]. Here, a Radial Based Function (RBF) Neural Network is implemented and applies on a UCLA dataset. The result attains 96% accuracy for DDoS attack detection.

A comparative analysis with some common approaches for statistical significance based feature selection is presented in [13]. The features of DoS attack are merged with Consistency-based Subset Evaluation (CSE) to determine significant features from the dataset. The CSE method calculates the inconsistency ratio for feature subsets by measuring inconsistency between the feature values. In [14], CAIDA'07 dataset is used and considered 16 features as most important. The authors propose an ensemble method to select features by considering the scores of statistical importance such as information gain, gain ratio, SVM, chi-square, relief, symmetrical uncertainty ranking filter and correlation ranking etc. They calculate the average from the individual score and set it as a threshold value for every feature either permit or block to enter in the final feature set. Granular computing technique together with entropy is used for feature selection in order to detect DoS attack is presented in [15]. The NSL-KDD'09 dataset is used and seven features are considered from the dataset with fifty instances in total. The authors calculate the entropy, counts the anomaly and assign weight value to each.

In [16], the authors consider HTTP flooding and simulate modern types of DDoS attack such as DDoS via SQL injection (SIDDOS) along with the traditional UDP flooding and Smurf attacks. The study compares among 27 features and applied several machine learning methods including Random forest, Multilayer Perceptron and Naïve Bayes for accuracy calculation. In another work, only 5 features are considered and C4.5 decision tree algorithm is applied to detect DDoS attack in cloud considering the splitting criteria of gain ratio [17]. But the training dataset of this study consists with a very limited number of features. Signature based detection method is used in the detection module to get better result. A performance comparison is also shown with two other classifiers, i.e. K-means and naive Bayes.

In [18], Random Forest algorithm is implemented on a customized dataset to detect DDoS attack. The scheme does not consider HTTP flood, a major type of application layer DDoS attack. Again, the detection of DDoS attack on application layer using feature construction and Logistic Regression is presented in [19]. This method only considers HTTP flood attack and projects high false positive rate. Additionally, deep learning based DDoS detection model are proposed in [20][21], where outdated KDDCUP99 and NSL-KDD dataset are chosen respectively. A performance comparison between Support Vector Machine (SVM) and Deep Feed Forward (DFF) is evaluated in [22]. but this work considers only one type of network layer DDoS attack. Another comparative analysis among naive bayes, decision tree and artificial neural network (ANN) based DDoS attack detection method is discussed in [23]. The result shows ANN achieves highest accuracy of 84.3%, which is not up to the mark. A precise DDoS attack detection result in cloud using SVM is achieved in [1]. However, the accuracy of detection degraded if two different types of attack occur simultaneously in the same cloud environment.

3. Experimental Setup

Now a day, attackers use advance type of DDoS attacks and most of them are network the application layer attacks. So, the traditional datasets that were used in most of the research for DDoS attack detection are not that kind of effective. New methods required to be practically assessed on a dataset containing recent and more advanced types of attacks like HTTP flooding. Moreover, other available data sets may include a lot of redundant and duplicate information which leads to an ultimate impractical result.

The simulation of DDoS attack in public cloud networks is illegal and is not a wise decision because it can affect the performance of the cloud service. Again, the public cloud (AWS, Azure, Oracle, Google Cloud etc.) also have applied different attack detection and prevention mechanism. So, due to resource limitation and both legal and ethical reasons, any kind of attack needs to be simulated in a virtual and sandboxed environment. In our research, for the experimental setup, we have deployed ownCloud [24], a free and open source cloud platform on top of VMware. This ownCloud uses Linux distribution. We deployed it on Ubuntu 16.04 OS with Apache web server, MySQL database and PHP 7.0.

3.1 Attack Generation

To generate DDoS flooding attack we use three tools namely: hping3 [25], mausezahn [26] and wreckuests [27]. A representation of DDoS attack on ownCloud is depicted in Fig. 1. We performed UDP flood attack, TCP SYN flood attack and ICMP flood attack by using hping3. For DNS flood attack we used mausezahn and for HTTP Flood attack wreckuests was used. We choose an attacking machine with the configuration - Windows 10 operating system, Intel corei5 2.67 GHz CPU with 8 GB RAM. While executing the DDoS attack, the above mentioned tools flooded the destination nodes with traffic.

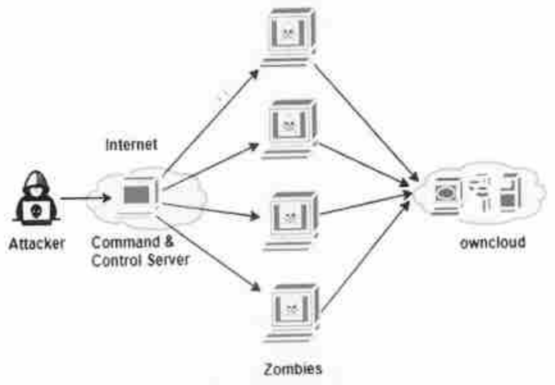


Fig. 1. Typical architecture of DDoS attack on ownCloud.

3.2 Dataset Collection

We use tcpdump [28], a traffic protocol analyzer to capture the attack traffic. Moreover, legitimate traffic was also collected using tcpdump from Lab environment network. The captured attack traffic and normal traffic were used to create a new dataset. The dataset has six classes and 1081633 records out of which 1001984 are DDoS attacks. Table 1 shows number of records of different classes of the dataset.

Then these data are pre-processed. In this step, redundant and duplicate records are removed. In order to apply feature selection method in the next step for identifying the most significant features, first we perform one-hot encoding to the "flag" attribute of the dataset and replaced it with the extracted specified flags. Table 2 lists all the features of the dataset we are dealing with.

Table 1: Distribution of Different Classes.

Types	No. of Records
TCP SYN flood attack	551179
ICMP flood attack	136496
UDP flood attack	125774
DNS flood attack	114160
HTTP flood attack	74375
Legitimate traffic	79649

Table 2: Extracted Data Set Features.

Sl	Features	Details
1	timestamp	Packet timestamp
2	protocol	Protocol name
3	length	Packet length
4	src_port	Source port number
5	dst_port	Destination port number
6	seq_no	Sequence number of the tcp packet
7	ack_no	Acknowledgement number of the tcp packet
8	win_size	Window size
9	len	TCP segment length
10	mss	Maximum segment size
11	ws	Window size
12	req_type	HTTP request type
13	resp_no	HTTP response number
14	query_resp	DNS query or response
15	req_reply	ICMP request or reply
16	ttl	Time to live
17	ACK	Acknowledge flag
18	FIN_ACK	Finish-Acknowledge flag
19	FIN_PSH_ACK	Finish-Push-Acknowledge flag
20	PSH_ACK	Push-Acknowledge flag
21	RST	Reset flag
22	RST_ACK	Reset-Acknowledge flag
23	SYN	Synchronize flag
24	SYN_ACK	Synchronize-Acknowledge flag
25	no_flag	No flag in packet

Next the features having categorical values are needed to be converted into numeric values. For this purpose, we assigned integer from 1 and onwards in every such feature, identified the distinct values for all the records in that column and then replaced with the assigned numeric values. Later the dataset is normalized in order to avoid any influence of features having high values over features containing low values. Our dataset collection process is illustrated in Fig. 2.

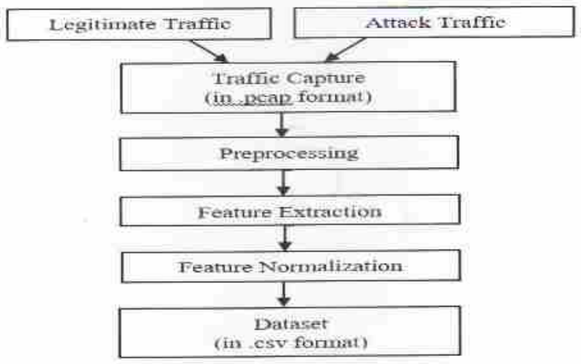


Fig. 2. Dataset collection steps.

4. Feature Engineering and MLA Selection

A block diagram of strategic-level framework for DDoS attack detection is depicted in Fig. 3. Feature selection is a significant step in the pattern recognition process, which defines the smallest possible set of variables capable of efficiently describing a set of classes [29]. Different methods for variable selection are discussed in the scientific studies and applied in software libraries as scikit-learn [30]. In this work, we have selected variables in two phases. First, Recursive Feature Elimination with Cross Validation (RFECV) is applied along with some widely used machine learning algorithms for classification problem in network and information security literature, i.e. Random Forest (RF), Decision Tree (DT), Naïve Bayes (NB), Logistic Regression (LR),and Stochastic Gradient Descent (SGD). RF achieved higher accuracy than others using 20 variables, while Stochastic Gradient Descent selected 7 variables, but achieved lower accuracy, as shown in Table 3.

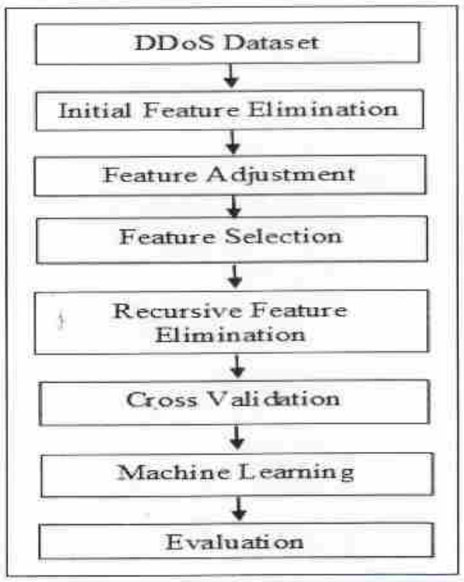


Fig. 3. A detailed framework for DDoS attack detection.

Table 3: 10-fold Recursive Feature Elimination with Cross Validation Results.

MLA	No. of Features	Accuracy
RF	20	0.991
LR	12	0.977
NB	12	0.975
SGD	7	0.902
DT	11	0.931

In the second phase, our proposed Algorithm 1 is applied for feature selection along with Random Forest algorithm. This proposed algorithm limits the number of variables from 20 to 17 for RF with a little climb in accuracy. Table 4 shows the result. To select the most relevant features, the feature importance for RF is calculated as mentioned in Algorithm 1. Fig. 4 illustrates the final output of our feature selection process.

Table 4: 10- fold Cross-Validation results for the selected variables.

MLA	No. of Features	Accuracy
RF	17	0.996
LR	17	0.989
NB	17	0.981
SGD	17	0.940
DT	17	0.932

Algorithm 1: Feature selection

```
Input: train features, train labels and number of rounds
Output: significant variables
begin
  Use the training set for training the model with all features
  Compute performance of the model
  Select the most significant variables from the trained model;
  for each subset size $S_i$ where i = 1…S do
      Keep the $S_i$ most significant features
      Train the model using $S_i$ features
      Test the trained model and compute the accuracy
  end for
  Calculate the performance profile over $S_i$
  Determine the proper number of features to use
  Use the RFE model returned by the optimum number of features $S_i$
  Calculate the accumulative significance of variables from the trained model
  Rank the variables by the mean of the calculated significance
  Return the "N" most significant variables;
end
```

Although RF utilises more variables than the other classifiers, it shows low false alarm rate and hence satisfies a primary precondition for DDoS attack detection system. So, RF becomes to be the best option for our machine learning based DDoS attack detection system. Random Forest is an ensemble learning algorithm mainly used for classification. It consists with many decision trees to create uncorrelated forests of trees and uses bagging method for training [31].

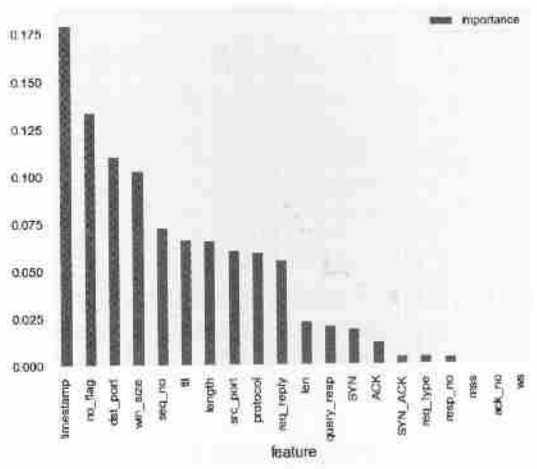


Fig. 4. Variables selection on the basis of the proposed feature selection algorithm.

5. Evaluation Metrics Calculation

Confusion matrix is generally used to evaluate the performance of the classifiers [32]. It visualizes the accuracy of a classification. In the confusion matrix shown in Table 5, TP means the number of true DDoS attack, TN means the number of true normal traffic, FP means the number of false DDoS attack and FN means the number of false normal traffic. These values are used to calculate the accuracy, precision, recall and f1-score of the classification as the primary performance indicators.

Table 5: Confusion Matrix.

		Predicted	
		Positive	**Negative**
True	**Positive**	TP	FN
	Negative	FP	TN

Accuracy: It represents the rate of the perfectly classified attack cases of both classes.

$$\text{Accuracy} = \frac{TP+TN}{TP+FN+FP+TN} \qquad (1)$$

Precision: It is the ratio of the number of positive instances retrieved to the total number of positive instances detected. It's another name is positive predictive. The equation of the precision is as follows:

$$\text{Precision} = \frac{TP}{TP+FP} \qquad (2)$$

Recall: It is the ratio of the number of positive instances retrieved to the total number of actual positive instances. It's another name is positive sensitivity value and can be calculated using the equation below.

$$\text{Recall} = \frac{TP}{TP+FN} \qquad (3)$$

f1-score: It represents how precise and robust the classifier is. So it is the harmony mean of precision and recall.

$$\text{f1-score} = \frac{2*Precision*Recall}{Precision+Recal} \qquad (4)$$

6. Results and Discussion

The evaluation of the classifiers is based on the values of the confusion matrix. The resultant confusion matrices for Random Forest, Logistic Regression, Naïve Bayes, Stochastic Gradient Descent and Decision Tree are represented in Tables 6, 7, 8, 9 and 10 respectively. We calculated the accuracy, precision, recall and f1-score of the models from the confusion matrices and listed in Table 11. The overall accuracy was 99.6%, 98.9%, 98.1%, 94% and 93.2% for Random Forest, Logistic Regression, Naïve Bayes, Stochastic Gradient Descent and Decision Tree correspondingly. Nevertheless, it is advisable that only the accuracy rate is not adequate, especially for imbalanced dataset just like our case. Therefore, the precision, recall and f1-score were calculated for each class.

Table 6: Confusion Matrix for Random Forest.

	DNS-flood	HTTP-flood	ICMP-flood	Legitimate traffic	TCP-SYN-flood	UDP-flood
DNS-flood	11284	0	8	0	0	0
HTTP-flood	0	7334	0	48	0	36
ICMP-flood	100	0	11586	0	0	0
Legitimate traffic	31	56	55	14604	0	32
TCP-SYN-flood	0	0	0	2	55117	0
UDP-flood	0	0	0	0	0	12510

Table 7: Confusion Matrix for Logistic Regression.

	DNS-flood	HTTP-flood	ICMP-flood	Legitimate traffic	TCP-SYN-flood	UDP-flood
DNS-flood	11284	0	8	0	0	0
HTTP-flood	0	6974	0	408	0	36
ICMP-flood	100	0	11586	0	0	0
Legitimate traffic	31	55	55	14604	1	32
TCP-SYN-flood	0	0	0	2	55117	0
UDP-flood	0	0	0	1	0	12500

Table 8: Confusion Matrix for Naïve Bayes.

	DNS-flood	HTTP-flood	ICMP-flood	Legitimate traffic	TCP-SYN-flood	UDP-flood
DNS-flood	11284	0	8	0	0	0
HTTP-flood	0	7382	0	0	0	36
ICMP-flood	100	0	11586	0	0	0
Legitimate traffic	815	307	60	13017	547	32
TCP-SYN-flood	0	0	0	3	55116	0
UDP-flood	0	0	0	0	0	12510

Table 9: Confusion Matrix for Stochastic Gradient Descent.

	DNS-flood	HTTP-flood	ICMP-flood	Legitimate traffic	TCP-SYN-flood	UDP-flood
DNS-flood	11280	4	8	0	0	0
HTTP-flood	1264	5099	0	1019	0	36
ICMP-flood	100	0	7457	4129	0	0
Legitimate traffic	38	41	55	14611	1	32
TCP-SYN-flood	0	0	0	2	55117	0
UDP-flood	0	0	0	0	0	12510

Table 10: Confusion Matrix for Decision Tree.

	DNS-flood	HTTP-flood	ICMP-flood	Legitimate traffic	TCP-SYN-flood	UDP-flood
DNS-flood	11284	0	8	0	0	0
HTTP-flood	0	143	0	7239	0	36
ICMP-flood	100	0	11586	0	0	0
Legitimate traffic	31	56	55	14604	0	32
TCP-SYN-flood	0	0	0	2	55117	0
UDP-flood	0	0	0	0	0	12510

Table 11: Performance of Algorithms.

Algorithms	Precision	Recall	f1-score	Accuracy
RF	0.99	0.99	0.99	0.996
LR	0.99	0.98	0.98	0.989
NB	0.98	0.97	0.98	0.981
SGD	0.95	0.94	0.94	0.940
DT	0.94	0.93	0.92	0.932

A comparative analysis on the precision, recall and f1-score between our considered five machine learning algorithms for detecting several DDoS attacks (in our case 5 types of DDoS attack) is also illustrated in Fig. 5, 6 and 7 respectively.

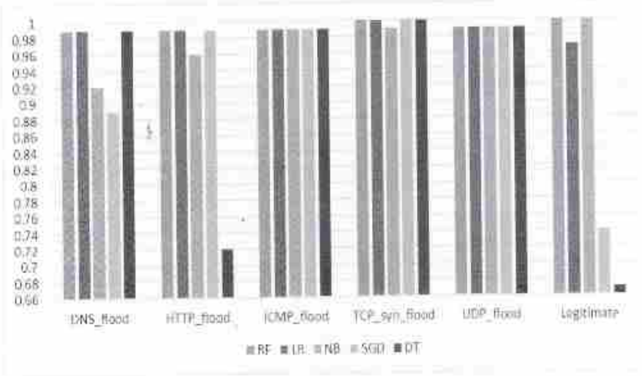


Fig. 5. Comparison of precision between the classifiers.

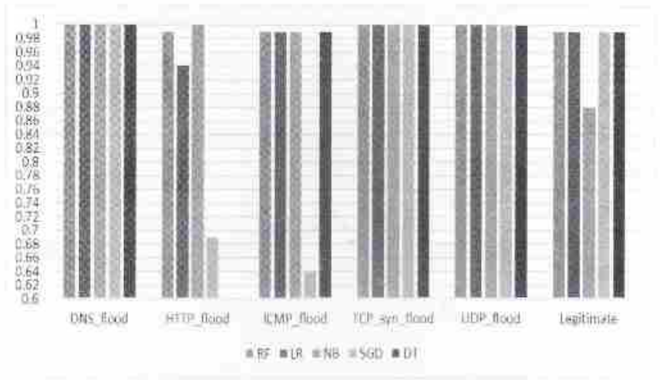


Fig. 6. Comparison of recall between the classifiers.

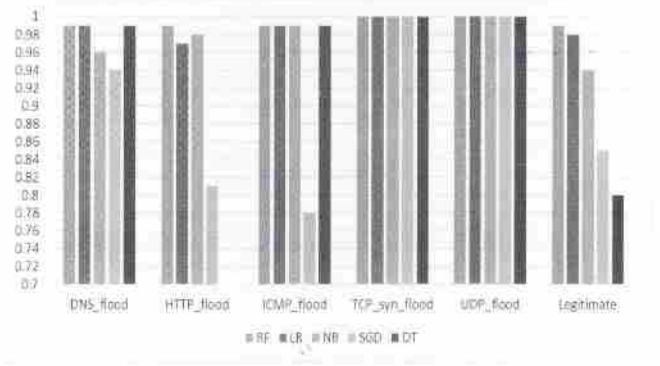


Fig. 7. Comparison of f1-score between the classifiers.

Among the five algorithms used, Random Forest represents the better results with regard to accuracy, precision, recall and f1-score. Logistic Regression also shows very similar result. Fig. 8 illustrates the comparison of the performance metrics between different machine learning classifiers.

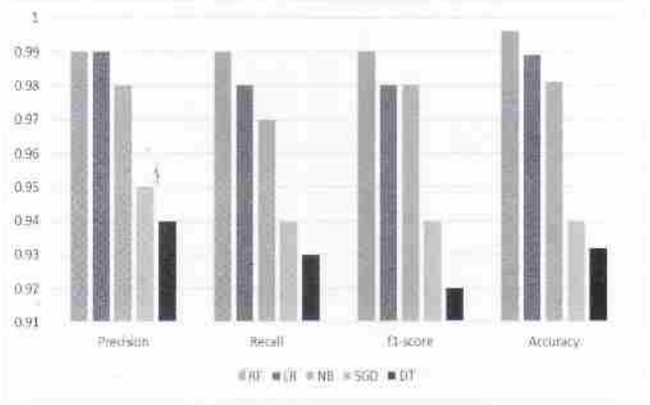


Fig. 8. Performance metrics comparison.

To eliminate the accuracy contradictory, the Receiver Operating Characteristic curve (ROC) is plotted and Area Under Curve (AUC) values shows actual accuracy of the model. Fig. 9 incorporates the ROC curves for all the classes for Random Forest. As we can see here, all the curves overlaps one another, so we have a clear distinction between the ROC curve and the base for all the six classes, as a result the maximum area between ROC curve and the base line is achieved here. Hence we have the AUC sore of 1 for the Random Forest.

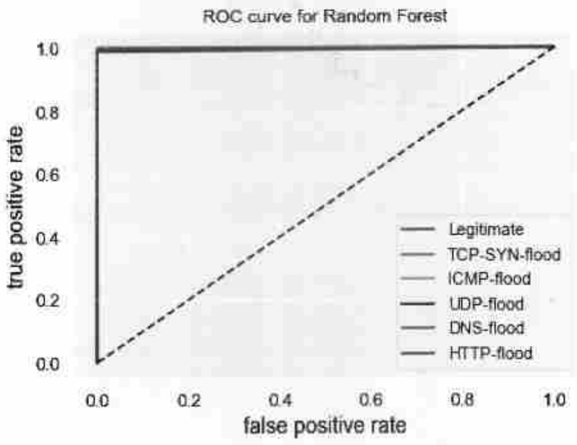


Fig. 9. ROC curves for Random Forest machine learning model for all classes.

To evaluate the effectiveness of our proposed method for detecting DDoS attack in cloud, we also make the comparison of the several key performance indicators for classification problems between our proposed method and seven other state of the art methods [33][21][22][23][17][34][35] as shown in Fig. 10. Form the figure, we can see that only Aamir [34] has a higher precision than our proposed method, but his methods has a very low recall value comparative to the others. For recall, f1-score and accuracy our proposed method has attained better results than the other methods.

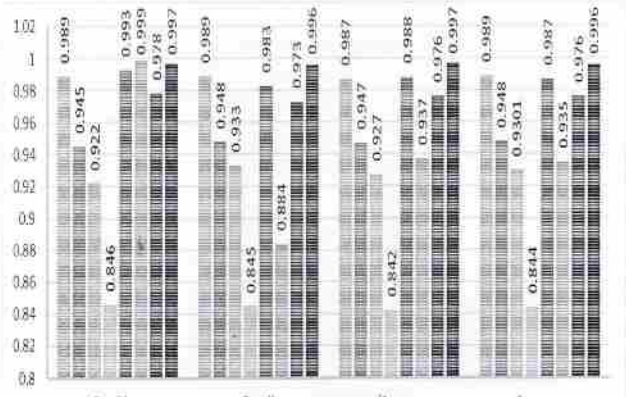


Fig. 10. Performance comparison with different state-of-the-art methods for DDoS attack detection.

7. Conclusion

In this paper, we have presented machine learning based DDoS detection system in cloud computing environment. In our work, we focused on network, transport and application layer DDoS attack in cloud. The proposed system is evaluated based on our generated dataset. We have applied RFECV and feature selection with our proposed algorithm in two steps to get the most important features in our dataset for the DDoS attack detection. The features that are selected in these steps are assessed with five classification algorithms. Each classifier has been trained and tested on the dataset and a comparative analysis between the machine learning classifiers is introduced. By evaluating the outcome, it can be concluded that the proposed approach, in which Random Forest algorithm is used, showed superior performance for the detection of the simulated DDoS attack in cloud computing environment with a 99.6% accuracy as well as higher scores in other key performance indicators in comparison with four others machine learning algorithms. Later, an analysis is made with AUC calculation of ROC curve to assess the optimized accuracies. Finally, the effectiveness of our proposed method to detect DDoS attack in cloud is shown by comparing seven other state of the art methods.

References

[1] H. Abbasi, N. Ezzati-Jivan, M. Bellaiche, C. Talhi, and M. R. Dagenais, "Machine Learning-Based EDoS Attack Detection Technique Using Execution Trace Analysis," *Journal of Hardware and Systems Security*, vol. 3, no. 2, pp. 164–176, June 2019.

[2] A. Praseed, and P. S.Thilagam, "DDoS Attacks at the Application Layer: Challenges and Research Perspectives for Safeguarding Web Applications," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 661 – 685, September 2018.

[3] S. Y. Nam, and T. Lee, "Memory-efficient IP filtering for countering DDoS attacks," in: Proc. of the 12th Asia-Pacific Network Operations and Management Conference on Management Enabling the Future Internet for Changing Business and New Computing Services, APNOMS'09, pp. 301–310, 2009.

[4] G. Maciá-Fernández, R. A. Rodríguez-Gómez, and J. E. Díaz-Verdejo, "Defense techniques for low-rate DoS attacks against application servers," *Computer Networks*, vol. 54. no.15, pp. 2711-2727, October, 2010.

[5] F. Salmen, P. G. Hernandes Jr, L. F. Carvalho, and M. L. Proenca Jr., "Using Firefly and Genetic Metaheuristics for Anomaly Detection based on Network Flows," in 11th Advanced International Conference on Telecommunications (AICT). June, 2015.

[6] M. Vijayalakshmi, S. M. Shalinie, and A. A. Pragash, "IP traceback system for network and application layer attacks," *Recent Trends In Information Technology (ICRTIT)*, April, 2012.

[7] A. Balobaid, W. Alawad, and H. Aljasim, "A study on the impacts of DoS and DDoS attacks on cloud and mitigation techniques," in International Conference on Computing, Analytics and Security Trends (CAST), IEEE, pp. 416-421, 2016.

[8] Z. He, T. Zhang, and R. B. Lee, "Machine Learning Based DDoS Attack Detection from Source Side in Cloud," in 4th International Conference on Cyber Security and Cloud Computing (CSCloud), IEEE, pp. 114-120, 2017.

[9] R. Ashfaq, A. Raza, X. Wang, J. Z. Huang, H. Abbas, and Y. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences 378*, pp. 484-497, 2017.

[10] Y. Gao, Y. Feng, J. Kawamoto, K. Sakurai, "A machine learning based approach for detecting DRDoS attacks and its performance evaluation," in 11th Asia Joint Conference on Information Security (AsiaJCIS), pp. 80–86, 2016.

[11] N.A. Singh, K.J. Singh, and T. De, "Distributed denial of service attack detection using Naive Bayes classifier through info gain feature selection," in International Conference on Informatics and Analytics, pp. 1-9, August 2016.

[12] R. Karimazad, and A. Faraahi, "An anomaly-based method for ddos attacks detection using rbf neural networks," in International Conference on Network and Electronics Engineering, IPCSIT, vol. 11, 2011.

[13] A. R. Yusof, N. I. Udzir, A. Selamat, H. Hamdan, and M. T. Abdullah, "Adaptive feature selection for denial of services (DoS) attack," in IEEE Conference on Application, Information and Network Security (AINS), pp. 81–84, 2017.

[14] K. J. Singh, and T. De, "Efficient classification of DDoS attacks using an ensemble feature selection algorithm," *Journal of Intelligent System*, Dec. 2017.

[15] S. Khan, A. Gani, A. W. A. Wahab, and P. K. Singh, "Feature selection of Denial-of-Service attacks using entropy and granular computing," *Arabian Journal for Science and Engineering (AJSE)*, vol. 43, no. 2, pp. 499–508, 2018.

[16] M. Alkasassbeh, G. Alnaymat, A. Hassanat, and A. Almseidin, "Detecting distributed denial of service attacks using data mining techniques," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 7, no. 1, January 2016.

[17] M. Zekri, S. E. Kafhali, N. Aboutabit, and Y. Saadi, "DDoS attack detection using machine learning techniques in cloud computing environments," in 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech), pp. 1-7, Feb. 2018.

[18] J. Pei, Y. Chen, and W. Ji, "A DDoS Attack Detection Method Based on Machine Learning," *Journal of Physics: Conference Series*, vol. 1237, Jun. 2019.

[19] S. Yadav, and S. Selvakumar, "Detection of Application Layer DDoS Attack by Modeling User Behavior Using Logistic Regression," in 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), Dec. 2015.

[20] J. Chen, Y. Yang, K. Hu, H. Zheng, and Z. Wang, "DAD-MCNN: DDoS Attack Detection via Multichannel CNN," in Proc. 11th International Conference on Machine Learning and Computing (ICMLC), pp. 484-488, Feb. 2019.

[21] A. S. Unal, and M. Hacibeyoglu, "Detection of DDOS Attacks in Network Traffic Using Deep Learning," in International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES), pp. 1-5, May, 2018.

[22] P. Khuphiran, P. Leelaprute, P. Uthayopas, K. Ichikawa, and W. Watanakeesuntorn, "Performance Comparison of Machine Learning Models for DDoS Attacks Detection," in 22nd International Computer Science and Engineering Conference (ICSEC), pp. 1-4, Nov. 2018.

[23] A. Sanmorino, "A study for DDOS attack classification method," in 1st International Conference on Advance and Scientific Innovation (ICASI), May, 2019.

[24] V. Karovic, M. Gregus, "Practical Implementation of Private Cloud Based on Open Source ownCloud for Small Teams - Case Study," in 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Nov, 2015.

[25] S. Sanfilippo, "Information Gathering," Feb. 2014. [Online]. Available: https://tools.kali.org/information-gathering/hping3.

[26] Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki /Mausezahn.

[27] Github, [Online]. Available: https://github.com/JamesJGoodwin /wreckuests.

[28] A. Wagoner, "A Beginners Guide to tcpdump," GSEC Practical v.1.4b, 2002.

[29] S. Ganapathy, K. Kulothungan, S. Muthurajkumar, M. Vijayalakshmi, P. Yogesh, and A. Kannan, "Intelligent feature selection and classification techniques for intrusion detection in networks: a survey," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, p. 271, 2013.

[30] J. Green, J. Juen, O. Fatemieh, R. Shankesi, D. Jin, and C. A. Gunter, "Reconstructing Hash Reversal based Proof of Work Schemes," in 4th USENIX conference on Large-scale exploits and emergent threats (LEET), p. 10, Mar. 2011.

[31] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[32] V. M. Patro, and M. R. Patra, "Augmenting weighted average with confusion matrix to enhance classification accuracy," *Transactions on Machine Learning and Artificial Intelligence*, vol. 2, no. 4, pp. 77 –91, 2014.

[33] Q. Li, L. Meng, Y. Zhang, J. Yan, "DDoS Attacks Detection Using Machine Learning Algorithms," *Communications in Computer and Information Science, Springer*, vol. 1009. 2019.

[34] M. Aamir, and S. M. A. Zaidi, "DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation," *International Journal of Information Security*, vol.18, pp. 761–785, Apr. 2019.
X. Yuan, C. Li, and X. Li, "DeepDefense: identifying DDoS attack via deep learning," in International Conference on Smart Computing (SMARTCOMP), pp. 1–8, 2017.
[35] X Yuan, C. Li, and X. Li, "DeepDefense: identifying DDoS attack via deep learning," in International Conference on Smart Computing (SMARTCOMP), pp. 1–8, 2017.

Authors Biography

Md. Mehedi Hasan is currently pursuing M.Sc. Engineering in Information and Communication Technology (ICT) from the Institute of Information and Communication Technology (IICT) of Bangladesh University of Engineering and Technology (BUET). He has worked as a Research Engineer in Asi@Connect Project titled "Distributed and Cloud-based Network Defense System for NRENs" at IICT, BUET. His research interest includes cyber security, machine learning, computer network and systems.

Md. Saiful Islam is a professor in the Institute of Information and Communication Technology (IICT) of Bangladesh University of Engineering and Technology (BUET). He obtained his B.Sc. in Electrical and Electronic Engineering from BUET, Dhaka in 1989; M. Sc. in Computer Science and Engineering from Shanghai University, China in 1997 and Ph. D in Electrical and Electronic Engineering degree from BUET in 2008.

Presently, he is serving as a Director of IICT, BUET. He has designed, coordinated and implemented various IT projects at national levels and published many articles in peer reviewed research journals. His research interest includes optical network, wireless communication, software engineering and cyber security. In his long academic career, he has supervised about 40 postgraduate students which results the solution of many real-life problems.

BCS Journal of Computer and Information Technology
Vol. 02, 2020, ISSN: 2664-4592 (Print), 2664-4606 (Online)

Discovering Interesting Knowledge from Transactional Database

Mohammad Fahim Arefin and Chowdhury Farhan Ahmed
Department of Computer Science and Engineering, University of Dhaka, Dhaka-1000, Bangladesh
Email s: f.arefin8@gmail.com, farhan@du.ac.bd

Abstract: Market basket data analysis and many other real-life applications store data in transactional database format, thus mining knowledge from Transactional Database has been a significant research topic. One aspect of interesting knowledge in transactional database is discovering homogeneous transactions. Existing algorithms measure the similarity between two transactions based on item co-occurrences, but none of them consider similarity between the set of items which appear in one of the two transactions. In this paper we propose an approach to mine interesting knowledge from transactional database based on homogeneity. Our measure can be used for segregating heterogeneous transactions and discovering object relationships in large datasets. We analyze the results obtained using our measure in real life datasets and explain how it discovers the inherent relationship and characteristics of data objects.

Keywords: Interesting Knowledge, Transactional Database and Data Mining.

1. Introduction

With the advent of computerized systems everywhere, terabytes of data are being generated every day. We need to process these data to extract useful knowledge which can later be used for multiple purposes and we use data mining to accomplish that. Data mining is the process of extracting interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from large information repositories such as: relational database, data warehouses, XML repository, etc. The term data mining is a misnomer, on the grounds that the objective is extracting information from raw data and changing it into a justifiable structure for further use, not the extraction (mining) of data itself. Pattern mining, association rule mining and correlation are three of the most active fields of research in data mining community. Frequent pattern mining and association rule mining[1] algorithms are used to get a generalized overview of data, but often this is not enough. Because when the minimum support threshold is high, most obvious knowledge will be found from the patterns and when the support threshold is low, a huge number of redundant patterns will be generated. We can use correlation analysis and similarity analysis to group similar association rules into a structure similar to multi-level association rules. [2] For some problems, we are more interested in finding patterns in small segments of data, instead of aggregate patterns. We can utilize transaction similarity to segregate transactions into homogeneous groups and then mine patterns or association rules in those groups for those cases.

Therefore transaction similarity can be useful in numerous scenarios. As transactions are comprised of items, their similarity should be an aggregate of the similarity between the comprising items. ROCK[3] first proposed that instead of typical distance metric based measures, using the number of common links to measure the similarity in categorical data will result in better results and this claim was further validated by other researchers in the past decade.[4]

There are numerous similarity measures for transactional data but the main drawback of existing measures is they calculate similarity between transactions based on only the number of items in common. They do not consider the similarity between mismatched items which results in inaccurate grouping of transactions.

To better identify similar patterns, we propose a similarity measure that takes into account two factors- number of common items and the similarity between the mismatched items. To make the measure robust and effective, we define it in such a way that it has three fundamental properties : null-invariant, normalized and symmetric. These properties ensure that the measure generates meaningful results without being biased by spurious data.

In this paper, our contribution is twofold. First, we propose an algorithm to calculate the similarity between two disjoint sets of transactions. Second, we propose a null-invariant and normalized symmetric measure to represent similarity between two transactions. We experimented on 6 real life datasets to validate our claim regarding the effectiveness of our proposed measure.

The rest of the paper is organized as follows. Section 2 defines the problem of measuring similarity in transactional database and reviews the previous works in this field. Section 3 describes the measure and algorithm for measuring transaction similarity. Section 4 presents the experimental study and section 5 presents the conclusions.

2. Literature Review

A metric function or distance function is a function that defines a distance between elements/objects of a set [5]. When measuring the similarity between two sets, we can use various distance measures like Euclidean distance, Manhattan distance, Minkowsky distance or similarity measures like cosine similarity etc. But in order to find similarity between two data points, distance based metrics calculate only the physical distance between two data points and hence, are inadequate when it comes to capturing the behaviour of transactional database. There is a high probability of the existence of similarity between two transactions even if they are far apart from each other as measured by the distance metrics[6].
Some of the most widely used similarity measures are discussed here in Table 1. For two sets A and B, we define the following notations,

$$M_{11} = A \cap B,$$
$$M_{10} = B - A,$$
$$M_{01} = A - B,$$
$$M_{00} = \{x : x \notin A \text{ and } x \notin B\}$$

Table 1: Existing similarity measures

Measure Name	Equation	Limitations
Simple matching co-efficient	$\frac{M11 + M00}{M11 + M00 + M01 + M10}$	Equal weight on M_{00} and M_{11}
Hamming distance	$A \oplus B = M01 + M10$	M_{11} is not considered
Jaccard similarity	$\frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert} = \frac{M11 + M00}{M11 + M01 + M10}$	Similarity between M_{01} and M_{10} not considered
Sorensen Coefficient	$\frac{2 * M11}{2 * M11 + M01 + M10}$	Similarity between M_{01} and M_{10} not considered
Cosine similarity	$\frac{A.B}{\lvert\lvert A \rvert\rvert * \lvert\lvert B \rvert\rvert}$	Similarity between M_{01} and M_{10} not considered

None of these measures are specific for transactions. ROCK[3] first proposed a measure for transactions, but it also counted the number of items two transactions had in common and QROCK[7] was just an optimized version of ROCK algorithm. Aggarwal[8] proposed the concept of measuring item affinity using jaccard similarity.

$$\boldsymbol{Affinity(i,j)} = \frac{\mathbf{sup}(\boldsymbol{i},\boldsymbol{j})}{\boldsymbol{sup}(\boldsymbol{i}) + \boldsymbol{sup}(\boldsymbol{j}) - \mathbf{sup}(\boldsymbol{i},\boldsymbol{j})}$$

This did not consider the similarity between the transactions where only i or only j appeared. The similarity between a pair of transactions T=i_1, ..., i_m and T'=j_1, ..., j_n was then defined to be the average affinity of their items. Formally,

$$\boldsymbol{Sim(T,T')} = \frac{\sum_{p=1}^{m}\sum_{q=1}^{n} \boldsymbol{A(i_p, i_q)}}{\boldsymbol{m.n}}$$

This work motivated us to find similar transactions in transactional database.

Table 2 : Example transaction database

Transaction	Items
T_1	A B C D E
T_2	G H I J E
T_3	A C G J E
T_4	A D H J E
T_5	A B C D F
T_6	G H I J F
T_7	A C G J F
T_8	A D H J F

$$TransactionSimilarity(i,j) = \frac{|C_{ij}| + TransactionMatch(M, N, Similarity\ Matrix)}{\frac{|Ti| + |Tj|}{2}}$$

Demonstration of Transaction Similarity: Let us suppose, we are considering two transactions T_3 and T_8 from Table 2:

T_3= A C G J E and T_8= A D H J F

We have two items in common between T_3 and T_8, therefore regular similarity measures will result in a 2/5 = 0.4 value. Now if we look at the items which appear in only one of the transactions, [C, G, E] in T_3 and [D, H , F] in T_8. We can observe that C-D, G-H and E-F are very similar items. Therefore we try to predict how similar is T_3 and T_8 using this information - they have 2 items in common and the other 3 items have a 80% similar counterpart in the other transaction.

$TransactionMatch([C,G,E],[D,H,F]) = 0.8 + 0.8 + 0.8 = 2.4$
Using these values, TransactionSimilarity = (2 + 2.4)/5 =0.88

4. Experimental Results

In this section, the performance study of the proposed algorithms will be evaluated under different performance metrics. Among the 7 datasets we experimented on 5 were collected from the spmf data mining repository [10] and the snake and UKRetail datasets were collected separately.

All the proposed algorithms were implemented in python and experiments were performed in Windows environment (Windows 10), on a 8th gen core-i7 intel processor which operates at 1.6GHz with 16 GB of memory. First, We demonstrate how we discover interesting knowledge in databases. Then we compare the runtime and memory complexity.

4.1 Similarity between Transactions

4.1.1 Foodmart

Foodmart is a sparse dataset containing 4141 customer transactions from a retail store, obtained and transformed from the SQL-Server 2000, having 1559 distinct items. In this dataset, we get a similarity value of 1 for 55 pairs of transactions, with a maximum length of 6.

Among the partially similar transactions, we get a maximum similarity of 0.78 between transaction 477 and 2402. Both of these transactions are of length 8 and they differ in only 1 item. Item 1170 appears in only Transaction 477 and item 756 appears only in Transaction 2402.

The two transactions are
Transaction 477: 1043 1365 1542 1426 217 727 1170 1399
Transaction 2402: 1043 1365 1542 1426 217 727 756 1399

As we can see they have 7 items in common with an average length of 8. Here the two distinct items are 1170 and 756, which represent two different product's productID.

4.1.2 UKRetail

This data set contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers. Here we have a large number of transactions which have a perfect similarity value which is expected from a retail shop.
As there are 4069 items and many 1 or 2 length transactions, 3148 transactions among the 5350 transactions are totally unique, that means they do not have even a single item in common. Among the partially similar transactions, we have a maximum similarity value of 0.9375 between transaction 3162 and 4900. Both of these transactions have Length 33 and contain 32 distinct items which are common to both of them. The only difference between these two transactions is Transaction 3162 has item 2872 twice and Transaction 4900 has it once. On the other hand, 1356 appears twice in Transaction 4900 and once in Transaction 3162.

Transaction 3162: 1356 2803 758 2 1795 2609 2611 1104 1095 1101 3882 1011 897 896 894 1648 527 526 594 3838 2873 2872 2870 2392 2458 2452 2469 2466 944 945 946 978 2872

Transaction 4900: 1356 2803 758 2 1795 2609 2611 1104 1095 1101 3882 1011 897 896 894 1648 527 526 594 3838 2873 2872 2870 2392 2458 2452 2469 2466 944 945 946 978 1356

We have sorted the transactions to easily demonstrate the similarity. We can see that the only difference is in the last item, this is why this pair has a very high similarity value.

4.1.3 *Sign*

For the sign dataset, we have a total of 266,085 pairs of transactions as there are 730 transactions. Among these we have 0 pairs with no similarity, we do not have any pairs in the 0.8 to 1 range and no transactions with a similarity of 1. So there are no two duplicate transactions.

This is supportive to our intuition that a dataset of sign language utterance will not have any duplicates and is expected to have some parts in common for most of the pairs.

We also ran our experiment in the snake dataset, mushroom dataset, chess dataset and the leviathan dataset. All these results, listed in Table 3 indicate that our similarity measure can discover similarity irrespective of the nature of the dataset.

4.2 Distribution over Similarity Ranges

The number of transaction pairs in each of the similarity ranges is an important characteristic of a dataset. We can analyze the inherent characteristics of a dataset based on the distribution of transaction pairs for different similarity ranges.

Fig. 1 represents the distribution of transactions for Foodmart, UkRetail and Snake Dataset. The X-axis represent the similarity ranges : negligible similarity between [0,001], similarity ranges with an interval of 0.2 and perfect similarity with similarity value 1 and in the Y-axis we represent the percentage of transaction pair which fall in a particular similarity range. Therefore a higher bar at a particular similarity range for a dataset means a larger portion of transaction pairs have similarity value which falls in that range.

As we can see from the figure, majority of the transaction pairs are in the range 0 to 0.4. Foodmart and UKRetail have thousands of items that is why almost all of the transaction pairs fall within the 0 to 0.2 range. In the foodmart dataset, among the 4141 transactions, we get around 17 million transaction pair combinations. As our measure is symmetric, we calculate around 8.5 million combinations and among those 8.46 million combinations have similarity value below 0.001. This is because it has very short transactions and around 1559 items. But in UkRetail, we have transactions of various length and there are around 84 thousand transactions with perfect similarity which is expected from a market basket data.

In the Snake dataset, most of the transaction pairs should be partially similar because it has a small number of items and it has very long transactions with just 20 items. This is clearly reflected in our analysis-it has most of the transaction pairs within the similarity range 0.001 to 0.4.

Similarly, **Fig. 2** represents the distribution of transactions for Chess, Mushroom and Sign dataset. We observe that majority of the transactions are in the range 0.2 to 0.8, because these 3 datasets have a small number of items, so it is expected that most of the transactions will share a number of items. Moreover, according to the nature of these datasets,there should not be many duplicates- this is also represented in our experiment as there is no duplicate for chess and sign dataset.

Table 3 : Number of transaction pair in each similarity range

Dataset	Items	[0, 0.001]	[0.001, 0.199]	[0.200, 0.399]	[0.400,0.599]	[0.600,0.799]	[0.800,0.999]	1.000
foodmart	1559	8,466,135	98,295	6,995	365	25	0	55
Snake	20	206	5,540	6,534	785	101	36	1
Sign	232	0	11,011	228,967	26,074	33	0	0
UKRetail	4,069	9,905,826	4,285,300	7,294	20,475	172	2	84,157
Mushroom	128	0	3,496,846	25,510,821	3,087,964	3,029,405	284,992	292
Chess	75	0	1	204,911	2,626,990	2,104,794	168,914	0
Leviathan	9,493	2,921,467	14,086,498	6,796	81	19	0	0

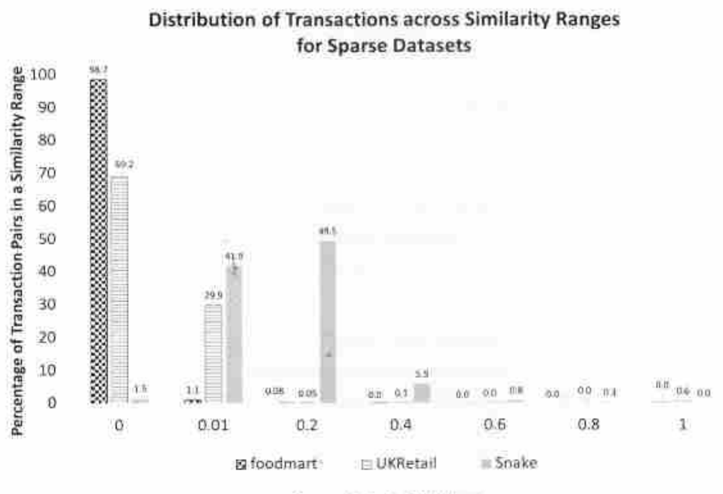


Fig. 1: Distribution of transaction pairs in sparse datasets

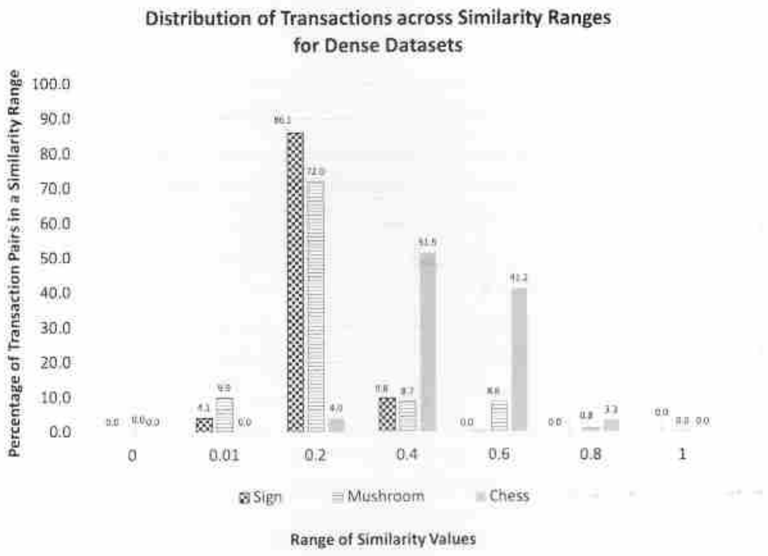


Fig. 2: Distribution of transaction pairs in dense datasets

Runtime Analysis: There are numerous algorithms to measure correlations and similarities between patterns. Our algorithm is not comparable to those because we do not have to mine patterns first to mine the correlations or similarities. The runtime of our algorithm mainly depends on the number of transactions in a dataset and the length of those transactions.

If we have N transactions in a dataset, generally, we have to calculate N*N similarities. But as our measure is symmetric, we have to analyze only half of those because Similarity(A,A)=1 and Similarity(A,B)=Similarity(B,A). Therefore, we calculate $\frac{n(n+1)}{2}$ similarities and the complexity of our algorithm is $O(n^2)$.

This is further reduced if we prune similarities below a certain thresholds. Then we can skip calculating similarities for uninteresting transaction combinations. We can apply multiple optimization techniques for this. Even without any optimization, our algorithm calculated 14 million transaction combination similarity in just around 200 seconds in a windows PC.

Memory Analysis: As our measure is symmetric we do not need to store the complete N*N matrix of similarities, we can store $\frac{n(n+1)}{2}$ similarities. The amount of memory required by our program depends on the average length of transactions, number of distinct items and number of transactions. For calculating 14 million transaction combinations in UKRetail, our algorithm took only 646MB of memory and this can be optimized further if we implement the memory manipulation in C++.

5. Conclusion

Measuring similarity in transactional database can result in the discovery of multiple dimensions of interesting knowledge. But existing measures do not measure the contextual similarity of items properly, because when comparing two items, they do not consider the similarity between the transactions where one of them appears. Thus we proposed an algorithm to measure similarity between two transactions. Measuring similarity between transactions in transactional database has numerous use cases. The experiments demonstrate that it can be calculated very quickly even for larger datasets and it is an effective measure across different contexts. We validate our claim with empirical results across real life datasets.

Acknowledgement

This research was partially supported by the "MS Fellowship" granted by "ICT Division, People's Republic of Bangladesh" to Mohammad Fahim Arefin.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499, San Francisco, CA, USA, 1994

[2] A. Y. Rodríguez-González, , J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, J. Ruiz-Shulcloper, & M. Alvarado-Mentado. Frequent similar pattern mining using non Boolean similarity functions. *Journal of Intelligent & Fuzzy Systems*, *36*(5), 4931-4944, 2019

[3] S. Guha, R. Rastogi and K. Shim. Rock: A robust clustering algorithm for categorical attributes. Information systems, vol. 25, no. 5, pp. 345–366, 2000

[4] K. McGarry, "A survey of interestingness measures for knowledge discovery," The knowledge engineering review, vol. 20, no. 1, pp. 39–61, 2005.

[5] J.P. Ortega, M. Del, R. B. Rojas, & M. J. Somodevilla, Research issues on k-means algorithm: An experimental trial using matlab. In CEUR workshop proceedings: semantic web and new technologies (pp. 83-96), 2009.

[6] A. Kulkarni, V. Tokekar and P. Kulkarni. Discovering context of labeled text documents using context similarity coefficient. Procedia comp. sci., vol. 49, pp. 118–127, 2015

[7] M. Dutta, A. K. Mahanta and A. K. Pujari. Qrock: A quick version of the rock algorithm for clustering of categorical data. Pattern Recognition Letters, vol. 26, no. 15, pp. 2364–2373, 2005

[8] C. C. Aggarwal, C. M. Procopiuc and P. S. Yu. Finding localized associations in market basket data. IEEE Transaction of Knowledge and Data Engineering, vol. 14, no.01, pp. 51–62, 2002

[9] T. Wu, Y. Chen and J. Han. Re-examination of interestingness measures in pattern mining: a unified framework. Data Mining and Knowledge Discovery 21, no. 3 (2010): 371-397, 2010.

[10] P. F. Viger, A. Gomariz, T. Gueniche, A. Soltani, C. Wu and V. S. Tseng. SPMF: a Java Open-Source Pattern Mining Library. Journal of Machine Learning Research, vol. 15, pp. 3389–3393, 2014.

Authors' Biography

Mohammad Fahim Arefin received the BSc degree in Computer Science from the University of Dhaka, Bangladesh in 2018 and currently completing his Masters from the same department. He received Azfar Ali Memorial Gold Medal from the President in the 51st Convocation of University of Dhaka for his academic performance in BSc. He also received the MS Fellowship from ICT Division, People's Republic of Bangladesh for this MS research. His research interests are in the areas of data mining, knowledge discovery and natural language processing.

Chowdhury Farhan Ahmed received the BS and MS degrees in Computer Science from the University of Dhaka, Bangladesh in 2000 and 2002 respectively, and the PhD degree in Computer Engineering from the Kyung Hee University, South Korea in 2010. He was a postdoctoral research fellow in the ICube Laboratory, University of Strasbourg, France from 2013 to 2015. He worked as a visiting scholar in the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia in March, 2019. Since 2004, he has been working as a faculty member (and as a Professor since 2016) in the Department of Computer Science and Engineering, University of Dhaka, Bangladesh. His research interests are in the areas of data mining, knowledge discovery and machine learning.

BCS Journal of Computer and Information Technology
Vol. 02, 2020, ISSN: 2664-4592 (Print), 2664-4606 (Online)

A Belief Rule Based Expert System to Evaluate Software Performance Under Uncertainty

Md. Mahashin Mia, Mohammad Shahadat Hossain and Rashed Mustafa
Department of Computer Science and Engineering,
University of Chittagong, Chittagong-4331, Bangladesh.
Emails: mahashin_csc@yahoo.com, hossain_ms@cu.ac.bd, rashed.m@cu.ac.bd

Abstract: The performance of software is disturbing in digitalization era, which has the ability to stop the everyday life activities of a digitalization area. Therefore, an earlier prediction of software performance could play an important role to save human times as well as daily life activities. The signs of reliability along with coverage and efficiency in system could be considered as a way to predict the software performance. These factors cannot be determined accurately because of the presence of various types of uncertainties. Therefore, this thesis presents a belief rule based expert system (BRBES) which has the capacity to calculate software performance under uncertainty. Chronological data of different software performance of the modern technology with specific reference to reliability as well as coverage and efficiency have been measured in validating the BRBES. The dependability of our projected BRBES's output is calculated in contrast with Artificial Neural Networks (ANN) based system and Fuzzy Logic Based Expert System (FLBES) , whereas our BRBES's results are found more dependable than that of ANN and FLBES . Therefore, this BRBES can be measured to calculate the incidence of a software performance in a area by taking account of the facts, related to the reliability, coverage and efficiency.

Keywords: Software, Uncertainty, Prediction, Expert system, Belief rule base.

1. Introduction

Most software performance problems and their complexity can be arised by the occurrence of the functional properties of the software. Most of the performance, which bring immense sufferings to the human work, can be noticed before their occurrence. Examples of such functional problems are Poor response time, Long Load time, Poor scalability, Bottlenecking and many others. It is fundamental to evaluate software performance since the early stages of the software lifecycle to reduce the risk of performance failures. In fact, experience shows that "performance problems are most often due to inappropriate architectural choices, rather than inefficient coding" [1].In extreme cases problems may be so severe to require considerable redesign and reimplementation, or even project failure [2]. Software performance evaluation is the process of predicting (early in the software development process) or assessing (towards the end of the development process) whether a software system is able to meet established performance objectives [3]. A software performance system can be defined using appropriate abstractions of the system structure and functions. The earliest explanation providing this information is Software Architecture (SA), defined as "the organization or structures of the system, which comprises software components, the superficially observable properties of those components, and the relations among them" [4].

Therefore, the prediction of software before its performances drew significant attention. In [5], certain criteria were suggested to identify the efficiency, compliance, and coverage of software. However, the prediction of software is recognized as yet to be solved problem of Computer science, although the performance of software features and computing resources has been investigated by the researchers of different countries for long time [6][7][8]. The performance and knowledge-based models have been widely preferred to predict software performance [7]. Queuing Network [9] Stochastic Process Algebra [10] and Stochastic Timed Petri Net[11] are the examples of performance models. In the performance model, different precursors are used to enable the short term prediction. Since the working environment is chaotic and complex, the short term prediction is inappropriate.

Table 1: Software performance uncertainty factors

Determinates	Uncertainty Types	Discussion
Response time	Incompleteness, vagueness	May refer to service requests in a variety of technologies.
Resource Utilization	Inconsistency, vagueness	The efficient and effective use of an organization's resources.
Time Recovery	Imprecision	Particular setting from a time in the past.
Accessibility	Imprecision	Brings benefits to everyone.
Availability	Imprecision	The probability that a system will work as required.
Velocity	Imprecision	The rate of change of position with respect to time.
Error rate	Imprecision	The rate at which errors occur in a transmission system.
Accuracy	Imprecision	Obtained from calculations depends on using bug-free computer chips.
Correctness	Ignorance, inconsistency	To set or make true, accurate or right.

However, the knowledge-based models used the prior information in predicting software performance. Software transferability [12] and Adaptive system characteristics [13] are the examples of such models. The other categories of this model consist of the approaches, which have been developed by using efficiency along with information related to the coverage and reliability to predict the software performance [14]. The accuracy of the prediction models depends on their capability of addressing various types of uncertainty, those exist with the signs of reliability as well as with the coverage and efficiency as illustrated in Table 1 [15].

An expert system can be thought of suitable substitute while there is an absence of algorithmic solution to a problem [16]. The software performance prediction is an instance of such a problem due to its involving complexity, multiple factors, often hard to calculate with accuracy. BRBESs (Belief Rule Based Expert Systems) are measured as the suitable candidates to apply in this type of complex problem [17]. Therefore, a BRBES with the capability of evaluating software performance by considering the reliability along with the coverage and efficiency is presented in this article.

The article is prepared in the following way. The present part introduces the significance of software performance and the possibility to predict such events. The literature review is enclosed in part II, while the BRBESs methodology is discussed in part III. Part IV presents our proposed BRBES to predict software performance. Results and discussions are elaborated in part V, while part VI concludes the article with an suggestion of future work.

2. Related work

Software Performance prediction is an important area of research, which is evident from the presence of various types of systems, available in the literature [6][7][8][9]. Graph transformation based methods, including UML(Unified Modelling Language) and LQN(Layered Queuing Networks) techniques are widely used to predict the software performance with high accuracy [18]. The LQN notation was developed as a mixture of Stochastic Rendezvous Networks and the Method of Layers presented in [19] to predict the software performance, which include software and hardware resources.

Extended Queuing Networks (EQN)[20] was also used to predict software performance. Software performance-related information using extensions defined in the "UML outline for Performance, Schedulability and Time" [21]. Generic process, based on the Performance of Software Engineering approach [22] described in [23]. In this system, Software performance assessment requires a systematic, inclusive process to distinguish the dynamic behaviour of a software system in quantitative terms. Architecture level software performance abstraction [24]. In this system, investigation of performance is restricted. Proxy Implementing Intelligence techniques [25].In this system, newly emerging research area. Expressiveness of performance [26]. In this system, Low Accuracy. Model-based software performance prediction. In this system, significantly evaluated the benefits and needs further validation. The performance of this system is better than that of human experts. However, relationship rule itself is a binary approach and therefore, uncertainty issues cannot be resolute by this approach [27].

Online survey, the non-functional requirements are measured as the important software performance prediction parameter [28]. The observation of activities of some parameter helps to predict software performance. The cause for this parameters have better perceiving control than person. The integration of AI methods such as expert systems could generate better calculation result in terms of correctness.

Hence, from the over it can be argued that all the software performance parameters as illustrated in Table 1 have not been considered by any of the systems in an integrated framework. Fuzzy logic based approaches are in capable of managing all categories of uncertainty having software performance elements as illustrated in Table I, especially ignorance, inconsistency and incompleteness both in the process of inference mechanisms and knowledge representation. On the contrary, BRBESs have the capability to represent the types of uncertainty as illustrated in Table 1 both in the inference processes as well as in the knowledge base [29][30] in an integrated framework. thus, the next part will introduce the BRBESs methodology.

3. Overview of Brbess Method

The BRBES's method represents uncertain knowledge, while it considers a few of steps in the inference procedure. This is elaborated below.

3.1 A schema to represent uncertain knowledge

Belief rules are used to represent uncertain knowledge, where a belief structure is used in the consequent part of each rule as shown in Eq. (1). Antecedent attributes are associated with the antecedent part with their referential categories as can be seen in Eq. (1). Thus, belief rules can be considered an up-gradation of classical IF-THEN rules.

$$\begin{aligned}
&R_K : IF(X_1 \text{ is } A_1^k) \cap (X_2 \text{ is } A_2^k) \cap \ldots\ldots \cap (X_{T_k} \text{ is } A_{T_k}^k) \\
&THEN \{(D_1 \text{ is } \beta_{1k}),(D_2 \text{ is } \beta_{2k}),\ldots\ldots,(D_N \text{ is } \beta_{nk})\} \\
&R_K : \beta_{jk} \geq 0, \left(\Sigma_{j=1}^{n} \beta_{jk} \leq 1\right) \\
&\textit{with a rule weight } \theta_k \\
&\textit{and attributes weights } \delta_1, \delta_2, \ldots, \delta_{Tk}, k \in 1,\ldots,L \qquad (1)
\end{aligned}$$

Where the Kth rule consists of Tk attributes in the left side of the rule. Each attribute of the left part of the Kth rule is associated with referential category. For example, AKi represents the referential category of the X1 attribute.

A rule is said to be complete if the summation of all the belief degrees related with each referential category of the consequent attribute of the attribute becomes "1". On the contrary, it is considered as incomplete.

A belief rule base comprises *L* rules. Fig. 1 represents a multilevel BRB framework, developed by taking the context of the software performance prediction parameters as shown in Table 1. This BRB framework consists of 4 BRBs, namely *D*10, *D*11, *D*12 and *D*13. The bottom level BRBs are D10, D11and D12, while the top level BRB is D13. The leaf nodes of D12BRB are the attributes of the antecedent part of the rules considered in this belief rule base, while D12 is the attribute of the consequent part. Eq. (2) can be used to compute the number of rules in D12BRB.

$$L = \prod_{i=1}^{T} J_i \qquad (2)$$

Where J_i is the referential categories related with antecedent attribute of a rule, while L denotes the number of rules available in a BRB.

If each leaf node of 'D12BRB contains three referential values, then by using Eq. (2), the value of L will become (3*3*3) = 27.

Eq. (3) illustrates the example of a rule associated with D13BRB.

From Eq. (3), it can be seen that belief degree 0% is embedded with "Low", 40% with "Medium" and 60% with "High".

3.2. BRBES's Inference Mechanism

3.2.1 Input Transformation

The value of an antecedent attribute can be transformed by finding its matching degrees to the referential values by using Eqs. (4) and (5) [29].

As "Correctness" ($D1$) is identified as "Low", then this linguistic variable is given a weight of 10% by an expert. Since the utility value for "Low" is measured as "0", for "Medium" as "50" and for "High" as "100" both in Eqs. (4) and (5), this weighted value 10% will be in the range of 50.

D1	Correctness
D2	Accuracy
D3	Error Rate
D4	Velocity
D5	Availability
D6	Accessibility
D7	Time Recovery
D8	Resource Utilization
D9	Response Time
D10	Efficiency
D11	Coverage
D12	Reliability
D13	Overall Performance

Fig. 1(a) Performance factors

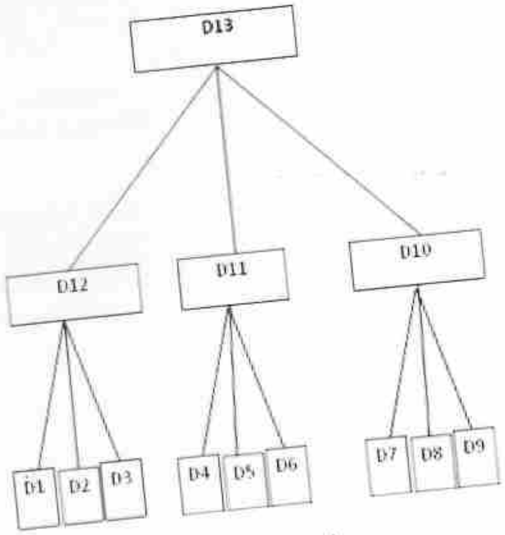


Fig. 1(b) BRB framework

Figure 1: The BRB framework for evaluating software performance

$$R_k : \left\{ \begin{array}{c} IF(Correctness\ is\ low)\ and\ (Accuracy\ is\ high)\ and \\ (Error\ Rate\ is\ Medium) \\ THEN \\ Software\ performance\ is \\ \{(High,(0.6)),((Medium,(0.4)),(Low(0.0))\} \end{array} \right. \quad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

$$IF(H\ value \geq Input\ value \geq M\ value)\ THEN \qquad (4)$$

$$Medium = \frac{H\ value - Input\ value}{H\ value - M\ value},$$

$$High = 1 - Medium,$$

$$Low = 0.0$$

$$IF(M\ value \geq Input\ value \geq L\ value)\ THEN \qquad (5)$$

$$Low = \frac{M\ value - Input\ value}{M\ value - L\ value},$$

$$Medium = 1 - Low,$$

$$High = 0.0$$

Therefore, in this case Eq. (5) will be applied, otherwise Eq. (4). Thus, by applying Eq. (5), the matching degrees for this input data (low) can be obtained for Low as 0.8 (Low = (50 10)/(50 0)= 0.8) for "Medium" as 0.2 (Medium = 1 0.8 = 0.2) and for High as "0", which are illustrated (see Table 2). When the referential categories are assigned with matching degrees then the rule is called packet antecedent and hence, it is considered as active.

3.2.2 Rule Activation Weight calculation

The activation weight calculation of a rule comprises calculating the combined matching degree, which is obtained by using Eq. (6) as well as by calculating activation weight, which is obtained by applying Eq. (7) [21].

Table2:Input transformation

Antecedent Name	Antecedent Value	Matching Degree	High	Medium	Low
Correctness (D1)	Low	10%	0.0	0.2	0.8

$$\alpha_k = \prod_{i=1}^{T_k} \left(\alpha_i^k\right)^{\delta_{ki}} \qquad (6)$$

where a$_k$ is the combined matching degree.

$$\omega_k = \frac{\theta_k \alpha_k}{\sum_{j=1}^{L} \theta_j \alpha_j} = \frac{\theta_k \prod_{i=1}^{T_k} \left(\alpha_i^k\right)^{\delta_{ki}}}{\sum_{j=1}^{L} \theta_j \left[\prod_{i=1}^{T_k} \left(\alpha_i^j\right)^{\delta_{ji}}\right]}, \delta'_{ki} = \frac{\delta_{ki}}{\max_{i=1,\ldots,T_k} \{\delta_{ki}\}} \qquad (7)$$

where d$_{ki}$ is the normalized antecedent attribute weight, obtained by dividing the individual antecedent attribute weight by the summation of all antecedent attribute weights of a rule. Hence, its value should be in between 0 to 1.

From Table 3, it can be observed that rule "6" consists of three antecedent attributes with their individual matching degrees, which need to be combined, by applying Eq. (6). The importance of this rule to calculate the Reliability can be acquired by applying Eq. (7). The implication of this value is that this rule has an important impact in getting the result or it is greatly perceptive.

Table 3: Rule activation weight calculation with combined matching degree

Rule Id	Antecedent			Consequent			Combined Matching Degree	Rule Activation Weight
	D1	D2	D3	H	M	L		
6	*L* (0.1)	M (0.6)	H (0.7)	0.1	0.6	0.3	0.015	1

3.2.3 Modified Belief Degree

This phenomenon can be considered as ignorance. In this state, the degree of belief of the original BRB needs to be modified, which can be obtained by using Eq. (8).

$$\beta_{ik} = \bar{\beta}_{ik} \frac{\sum_{t=1}^{T_k}\left(\tau(t,k)\sum_{j=1}^{J_i} \alpha_{ij}\right)}{\sum_{t=1}^{T_k} \tau(t,k)} \qquad (8)$$

Where

$$(t,k) = \begin{cases} 1, \text{if } P_i \text{ is used in defining } R_k\left(t = 1,\dots,T_k\right) \\ 0, \text{otherwise} \end{cases}$$

Here, $\bar{\beta}_{ik}$ is the initially assigned degree of belief, while β_{ik} is the modified degree of belief.

From Table 4, it can be observed that the original belief degrees of rule no. 6 have been modified since the input data of "Correctness" antecedent attribute is absent. The updated values of the belief degrees are obtained by applying Eq. (8).

Table 4: Belief Degrees Update

Rule ID		High	Medium	Low	Activation Weight
6	Initial	0.1	0.6	0.3	1
	Modified	0.09	0.34	0.6	0.095

3.2.4 Rule Aggregation

The rules of BRB need to be aggregated to obtain output data in response to the input data. As an instance, the input data of D12BRB consists of [D1=Low, D2 = Medium, and D3 = High]. The output value, i.e. the value of D12 consequent attribute, needs to be calculated in response to these input data, which can be achieved by aggregating the rules associated with D12BRB. The ER (Evidential Reasoning) inference mechanism is applied to obtain this overall calculative value in terms of fuzzy values. There are two forms of ER, namely, analytical and recursive. The analytical ER is measured to reduce the computational complexity as shown in Eqs. (9) and (10).

$$\beta_j = \frac{\mu \times \left[\prod_{k=1}^{L}\left(\omega_k \beta_{jk} + 1 - \omega_k \sum_{j=1}^{N} \beta_{jk}\right) - \prod_{k=1}^{L}\left(1 - \omega_k \sum_{j=1}^{N} \beta_{jk}\right)\right]}{1 - \mu \times \left[\prod_{k=1}^{L} 1 - \omega_k\right]} \qquad (9)$$

$$\mu = \left[\sum_{j=1}^{N} \prod_{k=1}^{L}\left(\omega_k \beta_{jk} + 1 - \omega_k \sum_{j=1}^{N} \beta_{jk}\right) - (N-1) \times \prod_{k=1}^{L}\left(1 - \omega_k \sum_{j=1}^{N} \beta_{jk}\right)\right]^{-1} \qquad (10)$$

where β_j illustrates the degree of belief related to the attribute of consequent referential category. By applying Eq. (9) for the input values of D12BRB, the calculated value for the consequent attribute "D12", which is obtained, consisting of (H, 0.3), (M, 0.7), (L, 0). By applying Eq. (11) the crisp value can be determined against the fuzzy values.

$$y_m = \sum_{n=1}^{N} D_n * \beta_n(m) \qquad (11)$$

where the expected numerical value is referred by Y_m, whereas each referential values's utility score is denoted by Dn. By considering the utility score for "High" as 10, for "Medium" as 7, and for "Low" as 0, the fuzzy values of D12 are converted into a numerical value, obtained as (10 * 0.09) + (7 * 0.34) + (0 * 0.6) = 3.28.

4. BRBES' to Evaluate Software Performance

4.1 BRBES' Design, Architecture and Implementation

The BRBES consists of a three-layer architecture, which comprises interface layer, inference procedures, and knowledge-base layers as can be seen in Fig. 2.

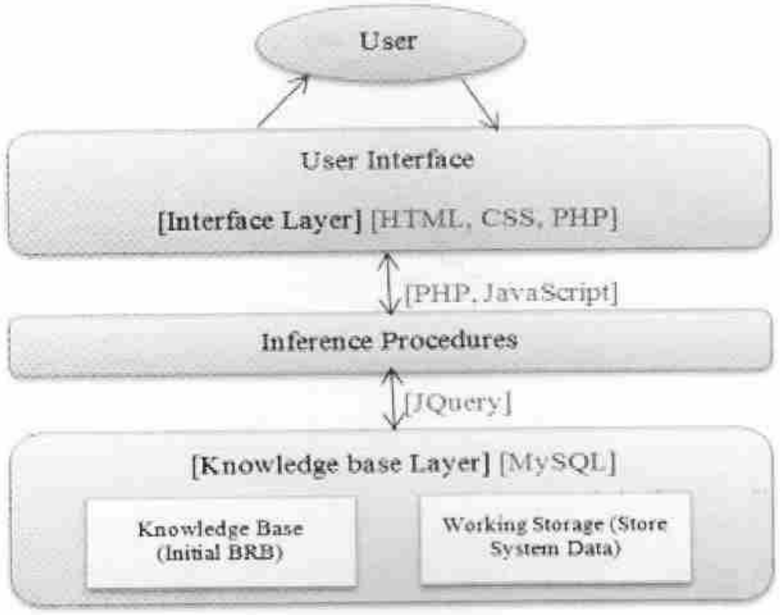

User
User Interface
[Interface Layer] [HTML, CSS, PHP]
[PHP, JavaScript]
Inference Procedures
[JQuery]
[Knowledge base Layer] [MySQL]
Knowledge Base (Initial BRB)
Working Storage (Store System Data)

Fig. 2: BRBES architecture

Since the structure is web-based, different web programming tools such as HTML, PHP and CSS are considered to construct the scheme interface. The inference level consists of different inference procedures of BRBES as discussed in the previous part. This layer has been developed by using JavaScript, PHP, and JQuery. For ease and shorter development cycles, PHP has been considered. To make the client side performance dynamic, and used JavaScript, which maintains the relation between the interface layers and inference. In contrast, JQuery has been considered to maintain the connection between the inference layers and knowledge-base. The BRBES's knowledge-base is developed using MySQL because of its flexibility. The preliminary BRB is also stored in MySQL. In addition, MySQL facilitates rapid data access and provides required security.

4.2 Knowledge Base Structure

The multi-level BRB structure is considered in consultation with domain experts. This structure is considered as the initial point to construct the knowledge-base. A BRB can be constructed by applying different approaches consisting of using knowledge of an expert, applying previous rules, examining previous data as well as creating casual rules. Here, attributes and rules are assumed to contain identical weight significance. "D12BRB" is illustrated in Table 5.

4.3 BRBES Interface

Fig. 3 shows the main interface of the scheme, while there are other interfaces to input data of the leaf nodes variables of Fig. 1 from the users. From Fig..3 it can be observed that for the certain input data of three leaf nodes (D1, D2, D3) of "D12BRB", the fuzzy values of the root node D12 i.e. "Correctness" have been obtained as (High, 73.6%), (Medium, 26.4%) and (Low, 0.00%). Using (11), this fuzzy value of D13 has been transformed into a numerical or crisp value, which is obtained as 85.027% as shown in Fig. 3. Here, one interesting finding is the fuzzy values of the mid-level nodes, which are "D10", "D11" and "D12" can also be converted into crisp values by using (11) and these can be used as the input data to the top level BRB, which is "D13BRB". In this way, a co-relation between the strength of software performance and the reliability, coverage and efficiency can be established by using this BRBES.

Table 5: Initial BRB for D12BRB

Rule Id	Rule Weight	IF Antecedent	THEN Consequent
		D1^D2^D3 is	A11 is
1	1	H^H^H	(H, 1.0), (M, 0.0), (L, 0.0)
2	1	H^H^M	(H, 0.0), (M, 1.0), (L, 0.0)
3	1	H^H^L	(H, 0.6), (M, 0.3), (L, 0.1)
4	1	H^M^H	(H, 0.5), (M, 0.4), (L, 0.1)
5	1	H^M^M	(H, 0.4), (M, 0.3), (L, 0.3)
6	1	H^M^L	(H, 0.3), (M, 0.2), (L, 0.5)
...	...	...	...
27	1	L^L^L	(H, 0.0), (M, 0.0), (L, 1.0)

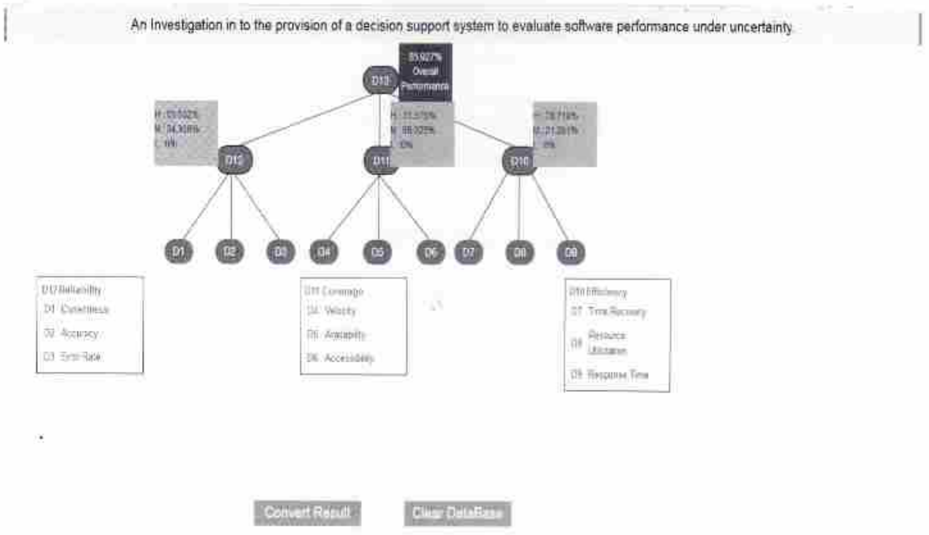

An Investigation in to the provision of a decision support system to evaluate software performance under uncertainty

Fig. 3: BRB interface

5. Results and Discussions

The dependability and the accuracy of the system has been determined by considering 125 datasets of different software around the technological world. These datasets are associated with the leaf nodes of the system structure as illustrated in Fig. 1. For simplicity, Table 6 illustrates datasets of 10 software, where columns 2–10 show the data of the leaf nodes, while column 11 shows the BRBES generated results in term of crisp value, Column 12 of Table 6 shows the original result of software performance. Fig. 4 illustrates the performance of software with a amount of 8.6, occurred, at DataSoft.com in Bangladesh. It is interesting to note that the BRBES generated software performance result for this software performance is 8.596, which is very close to the original software performance amount. BRBES's generated results were also compared with the Fuzzy Logic-based Expert System (FLBES). The output generated by FLBES for the same software is found as 8.19, which is far away from the software original data. In Table 6, column 13 illustrates the FLBES generated results.

Table 6: Software performance prediction results generated by BRBES and FLBES along with original results

Software performance (1)	D1 (2)	D2 (3)	D3 (4)	D4 (5)	D5 (6)	D6 (7)	D7 (8)	D8 (9)	D9 (10)	BRBES (11)	Original (12)	FLBES (13)	ANN (14)	Benchmark Data (15)
S1	90	90	70	80	75	80	75	90	75	7.6871	8.2	8.09	8.76	1
S2	80	85	80	70	70	70	80	75	70	6.9869	7.4	8.18	8.35	1
S3	90	95	75	85	70	75	75	80	85	8.596	8.6	8.19	8.11	1
S4	85	90	70	80	75	75	75	75	75	7.8138	7.7	8.08	7.88	1
S5	90	85	75	85	80	75	70	85	80	8.4402	8.4	8.11	8.36	1
S6	80	65	80	80	80	70	75	85	75	7.3517	7.0	8.10	8.35	1
S7	90	85	90	85	80	80	80	80	65	8.532	8.5	8.09	8.30	1
S8	80	80	80	85	80	80	80	85	80	7.792	8.2	8.07	8.33	1
S9	85	80	80	75	70	70	75	70	75	7.0237	6.8	8.06	7.10	0
S10	85	90	85	75	70	75	70	70	80	7.3381	7.5	8.09	8.40	1

Finally, an ANN based system was also developed. Its results are shown in column 14 of Table 6. The result of the same software by using this system is found as 8.11, which is also far away from the original data.

Table 7: Dependability comparison among four systems Area Under Curve

System		Asymptotic 95% Confidence Interval	
		Lower Bound	Upper Bound
Original	0.735	0.576	0.872
BRBES	0.979	0.941	1.000
FLBES	0.769	0.689	0.928
ANN	0.872	0.782	0.962

When a software performance with more than "6.8" is found then the benchmark value is considered as 1, otherwise it is considered as "0". Column 15 of Table 6 shows the benchmark data, which has also been used to generate ROC curves. SPSS 23 has been used to generate the ROC curves.

Fig. 5 illustrates the ROC curves representing a comparison of dependability among the BRBES, ANN, FLBES and the original data, obtained mainly by using the classical models. ROC curve with green line represents BRBES results while with purple line represents ANN; gray line represents FLBES while blue line represents original data. Table 7 illustrates the AUC for BRBES, ANN, FLBES and original data which are 0.979, 0.872, 0.769 and 0.735 respectively. Therefore, it can be argued that the dependability of software performance prediction of BRBES is better than that of original data because later obtained by using classical models which are not developed by taking account of different categories of uncertainty related with the different variables of software performance. On the contrary, the FLBESs only considers uncertainties due to vagueness, ambiguity and imprecision in their knowledge representation scheme. Therefore, the uncertainties due to randomness, ignorance and incompleteness, which are noticed in Table 1 with the software performance variables, are not considered in FLBES. On the other hand, the BRBES considers all categories of uncertainties associated with a knowledge representation schema and an inference mechanism which are found in Table 1. Thus, the BRBES's outputs are found reliable in comparison to FLBES as evident from Figure 5 and Table 7. Here, an interesting observation can be noticed that ANN based systems consider only one learning parameter i.e. weight, while BRBESs consisting of learning parameters rule weight, attribute weight and degree of belief [31]. Additionally, ANN represents black-box type of system, which is not concerned with the different types of uncertainties related with variables of software performance as illustrated in Table 1. Hence, ANN based system's outputs are not found reliable than from BRBES which can be seen from Table 7 and Fig. 5.

Fig. 4: Datasoft.com software performance, Bangladesh. [Original result: 8.6, BRBES results:8.596]

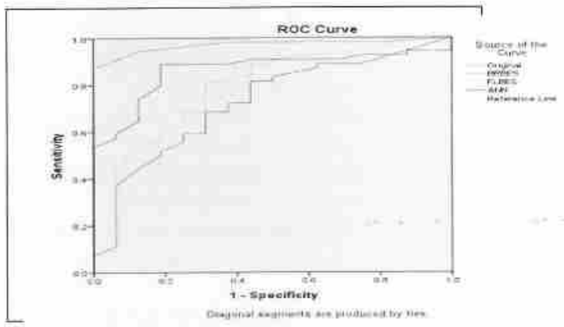


Fig. 5: ROC curves comparing dependability among BRBES ANN, FLBES and original result

6. Conclusion

The design, development as well as the applications of a BRBES to evaluate software performance from the reliability, coverage and from efficiency are presented throughout this article. A comparison of the BRBES's results with ANN, FLBES, and original data has been carried out. It can be noticed that BRBES' outputs are found dependable in comparison to ANN, FLBES and original data. As BRBES considers various categories of uncertainties related with the variables of reliability as well as with the coverage and efficiency. The BRBES, presented, in this paper is an example of a multilevel BRBES which allows the generation of various scenarios of software performance predictions. For example, the efficiency can be predicted alone before software performance occurrence. In the same way, both coverage and efficiency can be analyzed before software performance occurrence. In this way, the BRBES allows the investigation of possible software performance from a variety of perspectives and hence, the decision-makers could take appropriate measures to diminish the risk of software performance in a digitalization working area. Finally, by using the BRBES an aggregated calculative view of software performance amount can be obtained. Such a BRBES can easily be used to predict the software performance by looking at the reliability by anyone where there is a availability of Internet since the system is web-based. The real time software performance prediction could be Possible if the input data can be acquired by deploying wireless sensor network technologies in a region [32].

References

[1] L. G. Williams and C. U. Smith, "Pasa^{sm}: A method for the performance assessment of software architectures", Proc. of 3th ACM Workshop on Software and Performance WOSP'2002, pp. 307-320, July 2002.
[2] C. U. Smith and L. G. W illiams, *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*, Addison-Wesley, 2002.
[3] S. Balsamo, A. D. Marco, P. Inverardi, and M. Simeoni, "Model-based performance prediction in software development: A survey", *IEEE*
[4] L. Bass, P. Clements, and Katzman. *Software Architecture in Practice*. Addison Wesley, 1998.
[5] M. Woodside, *et al.*, "The Future of Software Performance Engineering," in Future of Software Engineering, 2007. *FOSE '07*, pp. 171-187, 2007.

[6] G. Gu, and D. Petriu, "From UML to LQN by XML algebra-based model transformation", Proc. of 5th ACM Workshop on Software and Performance, pp.99-110, July, 2005.

[7] D. C. Petriu, and X. Wang, "Deriving Software Performance Models from Architectural Patterns by Graph Transformations", *Lecture Notes in Computer Science* Vol. 1764, pp. 475-488, Springer, 2000.

[8] D. C. Petriu, and X. Wang, "From UML description of high-level software architecture to LQN performance models", *Lecture Notes in Computer* Science, vol. 1779, pp. 47-62, Springer, 2000.

[9] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley and Sons, 2001.

[10] G. Franks, *Performance Analysis of Distributed Server Systems*. PhD Thesis, Carleton University, Canada, 2000.

[11] F. Baccelli, G. Balbo, R. Boucherie, J. Campos, and G. Chiola, " Annotated bibliography on stochastic Petri nets", In *Performance Evaluation of Parallel and Distributed Systems-Solution Methods* (Amsterdam, 1994), C. Tract, Ed., no. 105, pp.1–24.

[12] J. A. McCall, *et al.*, "Factors in Software Quality," *Griffiths Air Force Base, N.Y. Rome Air Development Center Air Force Systems Command,* 1977.

[13] N. J. C. Primus, "A generic framework for evaluating Adaptive Educational Hypermedia authoring systems," MSc, Business Information Systems University of Twente, Enschede, 2005.

[14] *S.* Balsamo, A. D. I. Marco, P. Inverardi and Simeoni, "M. Model-based performance prediction in software development: A survey", *IEEE Transactions of Software Engineering*, vol. *30*, no. 5, pp. 295–310, 2004.

[15] C. E. de Barros Paes and C. M. Hirata, "RUP Extension For the Software Performance," in *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, 2008, pp. 732-738.

[16] S. Benferhat, A. Boudjelida, K. Tabia, and H. Drias, "An intrusion detection and alert correlation approach based on revising probabilistic classifiers using expert knowledge," Applied Intelligence, vol. 38, no. 4, pp. 520–540, June 2013.

[17] J.-B. Yang, J. Liu, J. Wang, H.-S. Sii, and H.-W. Wang, "Belief rule-base inference methodology using the evidential reasoning approach-rimer," IEEE Transactions on Systems, Man, and Cybernetics-part A: Systems and Humans, vol. 36, no. 2, pp. 266–285, March 2006.

[18] D. C. Petriu, and X. Wang, "Deriving Software Performance Models from Architectural Patterns by Graph Transformations", *Lecture Notes in Computer Science* Vol. 1764, pp. 475-488, Springer, 2000.

[19] Franks, G., Hubbard, A., Majumdar, S., Petriu, D., Rolia, J., and Woodside, C. A toolset for performance engineering and software design of client-server systems. *Performance Evaluation 24*, 1-2 (1995), 117–135.

[20] E. Lazowska, J. Kahorjan, G. S. Graham, and K. C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queuing Network Models*. Prentice-Hall, Inc., Englewood Cliffs, 1984.

[21] OMG. *UML Profile for Schedulability, Performance, and Time*. OMG document ptc/2002-03-02,http://www.omg.org/cgi-bin/doc?ptc/2002-03-02.

[22] C. U. Smith, *Performance Engineering of Software Systems*. Addison-Wesley, 1990.

[23] C. U. Smith, and Williams L. G. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*, Addison-Wesley, 2002.

[24] F. Brosig, N. Huber, S. Kounev "Architecture level software performance abstraction for online performance prediction" Elsevier(2013) Data management with attribute based encryption method for sensitive users in cloud computing. Vidyasagar Tella,L.V Ramesh.IJETR ISSN:2321-0869,vol-2,issue-9,September 2014

[25] S. Kounev, F. Brosig, N. Huber, and R. Reussner, "Towards self-aware performance and resource management in modern service-oriented systems", In: Proceedings of the 7th IEEE International Conference on Services Computing (SCC 2010),

[26] S. Becker, H. Koziolek, R. Reussner, "The palladio component model for model-driven performance prediction", Journal of Systems and Software, 82(2009) pp. 3-22.

[27] Y. Song, X. Wang, W. Wu, L. Lei and W. Quan, "Uncertainty measure for atanassov's intuitionistic fuzzy sets", Applied Intelligence, vol. 46, no. 4, pp. 757–7743, 2017.

[28] M. Atul, S. Mulik, and V. D. Thombre, "Survey Paper on Online Software Performance Prediction", International Journal of Advanced Engineering, Management and Science (IJAEMS), Vol-3, Issue-1, Jan- 2017.

[29] M. S. Hossain, S. Rahaman, A.L. Kor, K. Andersson, and C. Pattinson, "A belief rule based expert system for datacenter puc prediction under uncertainty," IEEE Transactions on Sustainable Computing, vol. 2, no. 2, pp. 140–153, April 2017.

[30] D.L. Xu, J. Liu, ian Bo Yang, G.-P. Liu, J. Wang, I. Jenkinson, and J. Ren, "Inference and learning methodol-ogy of belief-rule-based expert system for pipeline leak detection," Expert Systems with Applications, vol. 32, no. 1, pp. 103–113, January 2007.

[31] A. Bahrammirzaee, A. R. Ghatari, P. Ahmadi, and K. Madani, "Hybrid credit ranking intelligent system using expert system and artificial neural networks," Applied Intelligence, vol. 34, no. 1, pp. 28–46, February 2011.

[32] K. Andersson and M. S. Hossain, "Heterogeneous wireless sensor networks for flood prediction decision support systems," in Proc. of 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS'15), Hong Kong, China. IEEE, April 2015, pp. 133–137.

Authors' Biography

Md. Mahashin Mia is pursuing his M.Phil Degree in Computer Science and Engineering from University of Chittagong. After completing his B.Sc in 2006and M.Sc degree in Computer Science and Engineering from the same university in 2008, he is now Assistant controller of examination at Chattogram Veterinary and Animal Sciences University, Chattogram. His research interest includes the modeling of risks and uncertain-ties using evolutionary computing techniques, machine learning and the Internet of Things.

Mohammad Shahadat Hossain is a Professor of Computer Science and Engineering at the Chittagong University (CU), Bangladesh. He did both his MPhil and PhD in Computation from the University of Manchester Institute of Science and Technology (UMIST), UK in 1999 and 2002 respectively. His current research areas include e-government, the modeling of risks and uncertainties using evolutionary computing techniques. Investigation of pragmatic software development tools and methods, for information systems in general and for expert systems in particular are also his areas of research.

Dr. Rashed Mustafa is working as a professor in the department of Computer Science and Engineering at University of Chittagong. He is also acting as a dean of the faculty of engineering at University of Chittagong. His current research includes multimedia cloud computing, computer vision and high performance computing.

BCS Journal of Computer and Information Technology

Vol. 02, 2020 | ISSN: 2664-4592 (Print), 2664-4606 (Online)

Contents **Page**