Jordanian Journal of Computers and Information Technology

April 2019 | VOLUME 05 | NUMBER 01 | ISSN 2415 - 1076 (Online) ISSN 2413 - 9351 (Print)

JJCIT

www.jjcit.org jjcit@psut.edu.jo

An International Peer-Reviewed Scientific Journal
Financed by the Scientific Research Support Fund

Jordanian Journal of Computers and Information Technology (JJCIT)

The Jordanian Journal of Computers and Information Technology (JJCIT) is an international journal that publishes original, high-quality and cutting edge research papers on all aspects and technologies in ICT fields.

JJCIT is hosted by Princess Sumaya University for Technology (PSUT) and supported by the Scientific Research Support Fund in Jordan.

AIMS AND SCOPE

The JJCIT aims to publish the most current developments in the form of original articles as well as review articles in all areas of Telecommunications, Computer Engineering and Information Technology and make them available to researchers worldwide. The JJCIT focuses on topics including, but not limited to: Computer Engineering & Communication Networks, Computer Science & Information Systems and Information Technology and Applications.

INDEXING

JJCIT is indexed in:

EDITORIAL BOARD SUPPORT TEAM

JJCIT ADDRESS

WEBSITE: www.jjcit.org
EMAIL: jjcit@psut.edu.jo
ADDRESS: Princess Sumaya University for Technology, Khalil Saket Street, Al-Jubaiha
B.O. BOX: 1438 Amman 11941 Jordan
TELEPHONE: +962-6-5359949
FAX: +962-6-7295534

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 05, No. 01, April 2019.

DISTRIBUTED MUTUAL INTER-UNIT TEST METHOD FOR *D*-DIMENSIONAL MESH-CONNECTED MULTIPROCESSORS WITH ROUND-ROBIN COLLISION RESOLUTION

Jamil Al-Azzeh

(Received: 16-Oct.-2018, Revised: 23-Nov.-2018, Accepted: 8-Dec.-2018)

ABSTRACT

A collision-free extension to the mutual inter-unit test methodology for d-dimensional VLSI multiprocessors is proposed to guarantee that any processor core is tested only by its neighboring node at a time and no special care needs to be taken to choose those moments when test actions should start. Collision resolution hardware based on the round-robin arbitration routine is discussed in detail. A parallel collision-resolution-aware mutual inter-unit test algorithm is formulated and diagrammed. The proposed approach has been shown to improve the testability of mesh-connected multiprocessors by increasing the probability of successful fault detection as compared with the distributed self-checking methodology. Further, the new approach drastically reduces extra connectivity in the multiprocessor with respect to known mutual inter-unit test methods and leads to more easily manufactured multiprocessor fabric. For example, in a 4-dimensional system, we need 55% less extra connections with our approach.

KEYWORDS

Multiprocessors, VLSI, Mesh topology, Reliability, Testability, Self-test, Mutual inter-unit test.

1. INTRODUCTION

Continuing VLSI miniaturization has enabled the production of high-performance multicore and many-core single-chip multiprocessors comprising up to thousands of processor cores [4], [7]. However, the unreliability of such multiprocessor components has emerged as one of the crucial limitations to future scaling [12]-[13]. To maintain the correct operations of these multiprocessor systems with unhealthy components, specific fault-tolerance issues must be addressed when designing the multiprocessor [1], [3], [11], [15], [23], [27], [34] and [38]-[39]. Detecting the location of faulty components is one of these issues [2], [17] and [36].

A VLSI multiprocessor containing faulty components can be considered healthy if a dedicated fault detection and isolation mechanism is deployed [6], [9], [16], [21], [24], [31]-[32], [37]. With no specific mechanism of spare replacements, the multiprocessor maintains its operation, but its performance gradually degrades [19]-[20]. If a spare replacement mechanism is assumed [14], [18], the multiprocessor's performance is retained in the presence of the unhealthy components.

The detection of faulty components in VLSI multiprocessors is typically solved by built-in distributed self-checking or neighbor-checking methods [5], [10], [26], [28], [30], [40]. It is important that fault detection is done in-operation, which means that no long-term interruption of the multiprocessor is required to pinpoint an unhealthy component. Distributed self-test methods are an efficient, yet simple, solution to fault detection [8], [22], [25], [29], [33], [35]. However, these methods are characterized by relatively low testability: they may miss faulty components in some cases and/or treat healthy units as defective. Thus, the probability that a processor core is properly self-detected as faulty is not high enough for many practical applications. Another straightforward approach to fault detection is that a processor occasionally sends probe signals to its neighbors and marks neighbor cores as defective if no

J. Al-Azzeh is with Department of Computer Engineering, Al-Balqa Applied University, Al-Salt, Jordan. Email: jamil.azzeh@bau.edu.jo

acknowledgment is received within an established period of time. Such a neighbor-checking approach is also relatively simple to implement. However, it cannot provide better testability as compared to self-checking, because processor nodes test their peers independently and healthy/faulty decisions are made locally with no inter-processor cooperation.

A more complex fault detection mechanism, the mutual inter-unit test, has been specifically designed to improve multiprocessor testability by employing checking schemes with neighbor cooperation, by which each processor node is occasionally checked by a number of its neighbors and the final faulty/healthy decision is made according to the majority operator rule [41]. With this approach and depending on the topology of the multiprocessor, the probability of successful fault detection can be increased by at least 10% as compared to self-checking mechanisms. To improve the utilization of testing the hardware across a multiprocessor mesh and to make an additional increase in the probability of successful fault detection, a multiplexed mutual inter-unit test method based on a similar checking scheme to that of [41] and combined with distributed self-checking has been proposed [42]. Each test unit of each processor is now allowed to check a pair of its neighbors, A and B (not necessarily direct neighbors). The checking time period is split into two phases. During the first phase, neighbor A is tested while neighbor B is expected to send a test response. During the second phase, neighbor B is checked while neighbor A is expected to provide a test response. Thus, idle time is minimized and the testing hardware is used more efficiently in a time division manner. With this approach, the number of testing neighbors at each processing node is double of that in [41] with small hardware overhead, allowing a higher probability of successful fault detection. The main drawback of the multiplexed mutual inter-unit test is the assumption that each processor node (including those with non-direct processor cores) has many extra connections, which drastically increases the complexity of the communication network and may pose a serious problem if more dimensions are assumed. The inter-unit test methods presuppose that testing neighboring processor subsets check corresponding tested processors asynchronously, which may lead to collisions (two or more testing cores trying to check the same tested node at the same time); thus, special care must be taken to eliminate them.

Here, we propose another extension to the mutual inter-unit test methodology for *d*-dimensional VLSI multiprocessors by using a similar cooperating neighbor-checking scheme as that of [41]. Our main contribution is the use of a novel collision resolution mechanism (the round-robin collision resolution scheme), which guarantees that any processor core is tested by only one neighbor node at a time, so that no special care need be taken to choose the moments when test actions should start. In the following sections, we formally state the proposed method for a *d*-dimensional VLSI multiprocessor to concurrently detect faulty/defective nodes across a mesh. A parallel inter-unit test algorithm based on the proposed formal approach is presented and the dedicated test hardware implementing the above algorithm is diagrammed and briefly discussed. We also take a closer look at the round-robin collision resolution scheme, which is the cornerstone of our method. At the end of the paper, we compare our approach to the distributed self-checking technique and existing inter-unit test methods.

2. MUTUAL INTER-UNIT TEST AND COLLISION RESOLUTION FUNDAMENTALS

The idea of the mutual inter-unit test is straightforward. Each processor core is occasionally checked by its neighbors (referred to as “testing neighbors”). It is also assumed that a processor core periodically performs self-testing. The faulty/healthy decision for each processor is made according to the majority operator rule applied to the individual faulty/healthy decisions arriving from the testing neighbors and self-test hardware. The set of testing neighbors for each processor is formed according to the number of dimensions (*d*) of the multiprocessor topology, whose cardinality should be odd to make the majority operator applicable. The mutual inter-unit test procedure is carried out concurrently across the mesh, so that the faulty processors are detected and the corresponding signals are immediately transferred to the physical neighbors in order to isolate the faulty/defective cores in a timely manner. Unlike the distributed self-checking and neighbor-checking methods, the mutual inter-unit test mechanism provides for the operability of the test hardware itself to be tested implicitly. For example, if one of the testing processors issues a wrong faulty/healthy decision for its neighbor, then the tested neighbor (which is, in fact, healthy) will not be assumed as faulty by mistake, because the resulting faulty signal is formed by the majority operator. Therefore, the probability of successful fault detection increases.

The problem with the inter-unit test mechanism is that collisions may occur when several testing

neighbors start checking the same tested processor at the same time. Known inter-unit test schemes do not introduce any dedicated procedures and hardware to resolve the collisions; so, extra software-level solutions are necessary to calculate the time windows when a processor is allowed to test its neighbors with no collisions. However, this may drastically slow down the test process across the mesh, but may work against the multiprocessor's reliability. Hence, we propose an extended version of the inter-unit test with no possible inter-processor collisions. We assume that a hardware-level collision resolution mechanism, which we refer to as *the round-robin arbitration scheme*, is added to each processor core.

The idea of the collision resolution mechanism is for each testing neighbor (including the self-test units) of a given processor to be assigned an arbitration flag whose high value grants permission to start the test procedure. If this flag is zero, then the corresponding testing neighbor is not permitted to perform the test. The set of arbitration flags of each tested neighbor are organized as a ring shift register containing only the high value, which moves along the ring in a given direction and activates only one testing neighbor at a time. When a testing neighbor is about to initiate the test, it first polls the arbitration flag. If the flag is high, then the testing neighbor commences the test procedure and the flag stops moving until the test is finalized. If the flag is low, then the test is not initiated. The testing neighbor may be put into a queue to spin until the flag clears or simply leaves and tries to initiate the test the next time. With such an arbitration scheme, each processor is guaranteed to be tested by only one neighbor (or self-test unit) at a time. Therefore, no extra software-level support is required to pre-determine the time for initiating the test.

The remainder of the paper is organized as follows. In Section 3, we formally define the construction rule of the testing neighbor sets for a *d*-dimensional multiprocessor. Section 4 provides details on the proposed inter-unit test procedure. In Section 5, a hardware-level implementation of our approach is discussed. Section 6 provides the necessary details on the round-robin arbitration scheme. Section 7 is dedicated to the evaluation and comparison of our proposed approach to the distributed self-test and existing inter-unit test solutions. Section 8 contains the concluding remarks.

3. THE FORMATION OF TESTING NEIGHBOR SETS

We propose a more straightforward rule to form testing neighbor sets for each processor as compared to those defined in [41] and [42]. We assume that only the direct neighbors of a given core can be its testing neighbors, thereby eliminating extra diagonal connections among the processors and reducing the communication hardware complexity of the multiprocessor. With 4 direct neighbors in a 2-dimensional mesh, each processor has 5 testing neighbors if self-checking capabilities are assumed. With 6 direct neighbors in a 3-dimensional mesh, there will be 7 testing neighbors at each processor node. Analogously, in a *d*-dimensional mesh, each processor will be checked by $2d + 1$ testing neighbors.

Taking into account the processor nodes at the edges of the mesh, we can formally state the above rule as follows. Let us first consider a 2-dimensional multiprocessor. Let $U = \{u_{xy}\}$ be the set of its processors with *x* and *y* standing for the coordinates of a processor relative to the leftmost and lowermost node of the mesh, $x = \overline{0, n-1}$ and $y = \overline{0, m-1}$, respectively. *m* and *n* denote the numbers of rows and columns, respectively, of the mesh structure. Then, the testing neighbor set K'_{xy} of processor u_{xy}, $x \in \{0,1,\dots,n-1\}$, $y = \{0,1,\dots,m-1\}$, will be formalized as:

$$K'_{xy} = K_{xy} \cup \{u_{xy}\}, \tag{1}$$

$$K_{xy} = \left\{ u_{x,(y+1) \bmod m}, u_{(x+1) \bmod n, y}, u_{x+(1-\mathrm{sign}(x))n-1, y}, u_{x, y+(1-\mathrm{sign}(y))m-1} \right\}. \tag{2}$$

In Figure 1, different allocations of testing neighbors for the 2-dimensional case are illustrated. The testing neighbors are shown in grey and the dotted squares denote the testing nodes (which are mapped onto the corresponding cores at the opposite sides of the mesh) missing at the edges of the mesh.

Rules (1) and (2) can be directly expanded into a *d*-dimensional multiprocessor case:

$$K'_{x_1 x_2 \dots x_d} = K_{x_1 x_2 \dots x_d} \cup \left\{ u_{x_1 x_2 \dots x_d} \right\} \tag{3}$$

$$K_{x_1x_2\ldots x_d}=\left\{\begin{matrix} u_{(x_1+1)\bmod n_1,x_2,\ldots,x_d}, u_{x_1,(x_2+1)\bmod n_2,x_3,\ldots,x_d},\ldots,u_{x_1,x_2,\ldots,(x_d+1)\bmod n_d}, \\ u_{x_1+(1-\mathrm{sign}(x))n_1-1,x_2,\ldots,x_d}, u_{x_1,x_2+(1-\mathrm{sign}(y))n_2-1,x_3,\ldots,x_d},\ldots,u_{x_1,x_2,\ldots,x_d+(1-\mathrm{sign}(y))n_d-1} \end{matrix}\right\}. \quad (4)$$

It is evident that

$$\left|K_{x_1x_2\ldots x_d}\right|=2d\,, \quad (5)$$

$$\left|K'_{x_1x_2\ldots x_d}\right|=2d+1\,. \quad (6)$$

Formula (6) guarantees an odd number of testing neighbors at each processor and renders the majority operator applicable to faulty/healthy decisions. The collision resolution mechanism, in turn, guarantees that any processor u_{xy} is never checked by more than one peer at a time.

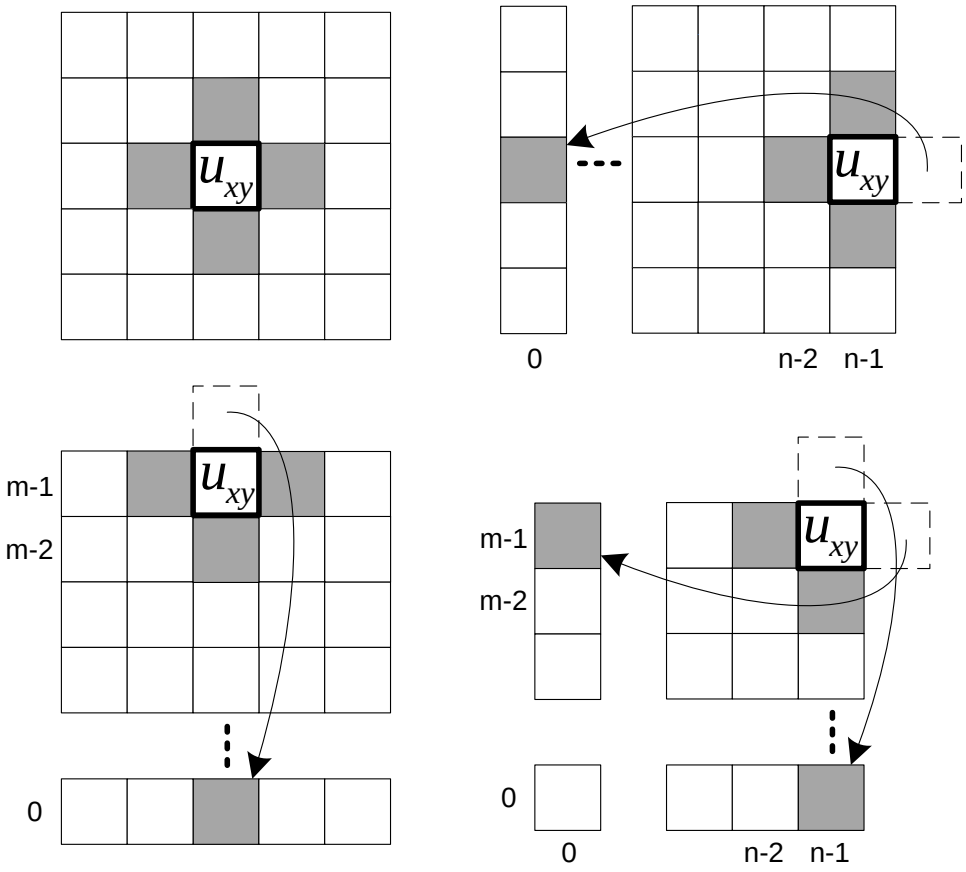


Figure 1. Possible allocations of testing neighbors K_{xy} in a 2-dimensional multiprocessor.

4. MUTUAL INTER-UNIT TEST PROCEDURE

The above conceptual description and formal rules allow the creation of an algorithm representing the process of the inter-unit test occasionally performed by each processor as we explain in detail below.

We consider a *d*-dimensional multiprocessor and its arbitrary processor $u_{x_1x_2\ldots x_d}$. For clarity, we indicate $u_{x_1x_2\ldots x_d}$. with the superscript 0 and enumerate its consecutive neighbors by the superscripts$1,2,\ldots,2d$, respectively. For the 2-dimensional case, we would have 4 neighbors renumbered as $u^1_{xy}, u^2_{xy}, u^3_{xy}, u^4_{xy}$.

The proposed algorithm is presented in Figure 2. All the symbols used in the flow-chart are explained in Table 1.

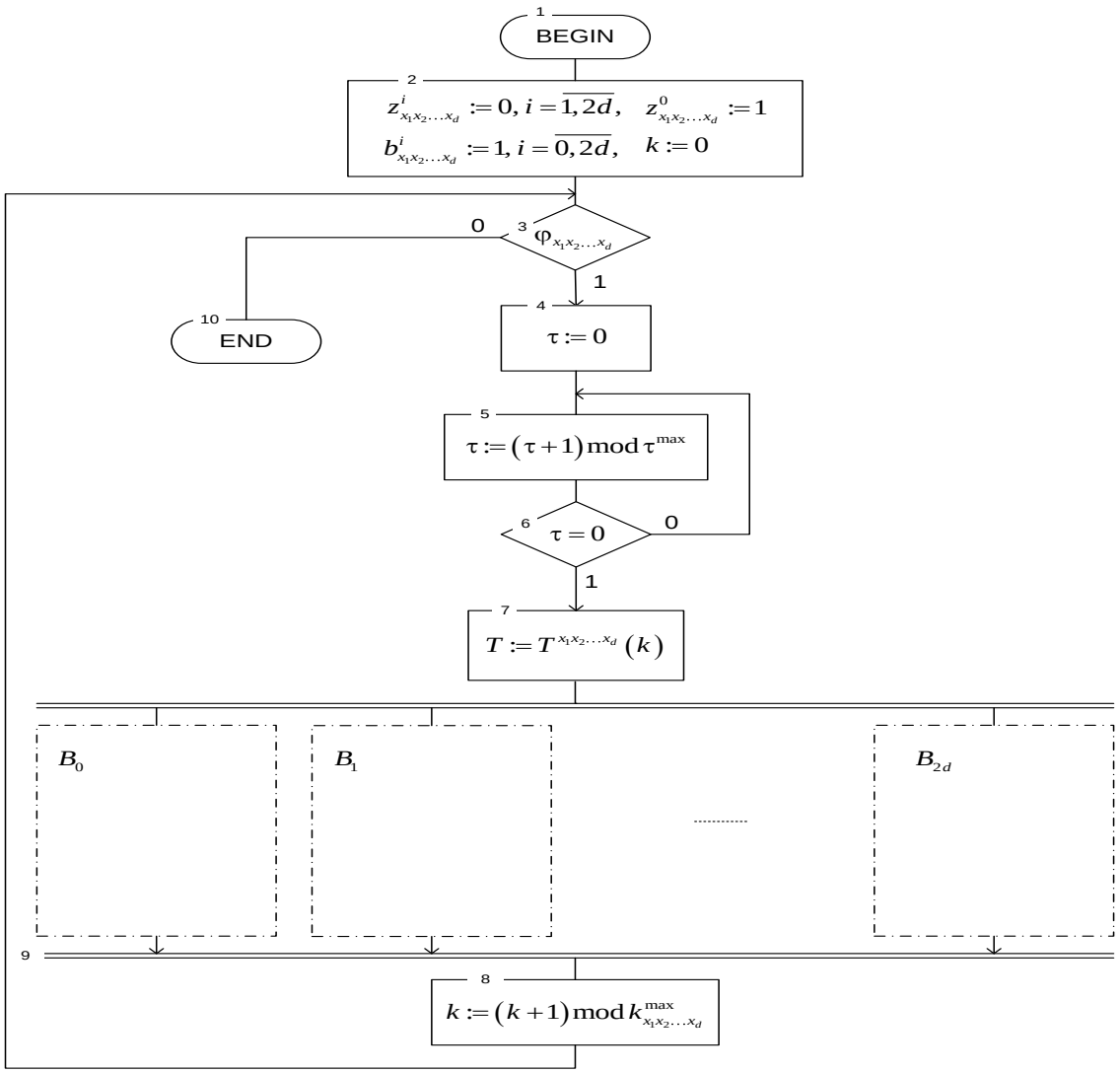


Figure 2. The proposed mutual inter-unit test algorithm.

Table 1. Symbols used in the flowchart shown in Figure 2.

No.	Symbol	Meaning
1	$T^{x_1x_2\ldots x_d}(k)$	k^{th} test signature issued by processor $u_{x_1x_2\ldots x_d}$
2	$k,\ k = \overline{0, k^{\max}_{x_1x_2\ldots x_d} - 1}$	Test signature counter
3	T	Current test signature to be transferred to the tested neighbors
4	$k^{\max}_{x_1x_2\ldots x_d}$	Number of test signatures supported by node $u_{x_1x_2\ldots x_d}$
5	τ	Test loop start timer
6	$\tau^{\max}$	Time between two adjacent test loops (in clock ticks)
7	$z^i_{x_1x_2\ldots x_d}$	Test enable flag for testing node $u_{x_1x_2\ldots x_d}$ and tested node $u^i_{x_1x_2\ldots x_d}$
8	$b^i_{x_1x_2\ldots x_d}$	Test enable flag for tested node $u^i_{x_1x_2\ldots x_d}$ and other testing neighbors (not including $u_{x_1x_2\ldots x_d}$)
9	$\varphi_{x_1x_2\ldots x_d}$	Healthy/faulty flag of node $u_{x_1x_2\ldots x_d}$
10	$B_0, B_1, \ldots, B_{2d}$	Separate parallel test threads corresponding to the tested neighbors and the self-test hardware of node $u^i_{x_1x_2\ldots x_d}$
11	:=	Assignment/transfer operator

The algorithm contains an outermost loop, including a parallel section and commences execution with the initialization (see Statement 2) to set up the collision resolution flags properly. Flags $z^{i}_{x_1x_2\ldots x_d}$ are reset to zero, which means that no neighbor of the current processor is allowed to start the test routine. In contrast, flag $z^{0}_{x_1x_2\ldots x_d}$ is set to logical "1," which means that it is allowed to perform the self-test.

As soon as the initialization ends, the algorithm enters the loop and executes until the current node is assumed to be healthy (see Condition 3). When the next iteration begins, the test loop timer starts counting down first (see Vertices 4–6). As soon as the timer has finished (the predefined time interval τ^{max} has elapsed), the next test signature $T^{x_1x_2\ldots x_d}(k)$ is fetched to initiate the test actions in the tested neighbors of the current node. A test signature may be interpreted as a pointer (address) to the test routine to be executed. All test routines are assumed to have been predefined and distributed among the processor cores in advance.

As soon as test signature $T^{x_1x_2\ldots x_d}(k)$ is read out, the algorithm enters the parallel section and threads $B_0, B_1, \ldots, B_{2d}$ to start the execution (see dashed–dotted squares in Figure 2). All these threads are identical. Thread B_i corresponds to the i^{th} tested neighbor and thread B_0 is mapped onto the current node (self-test). When all the threads terminate, Statement 8 executes to increment the test signature counter k and the next iteration begins. As soon as all $k^{\max}_{x_1x_2\ldots x_d}$ test signatures are fetched and processed, the algorithms roll back to test signature $T^{x_1x_2\ldots x_d}(0)$, assuming $k = 0$.

Each thread B_i $(i = \overline{0,2d})$ can be represented as a separate algorithm, as shown in Figure 3. All the symbols used in the flow-chart of Figure 3 are explained in Table 2.

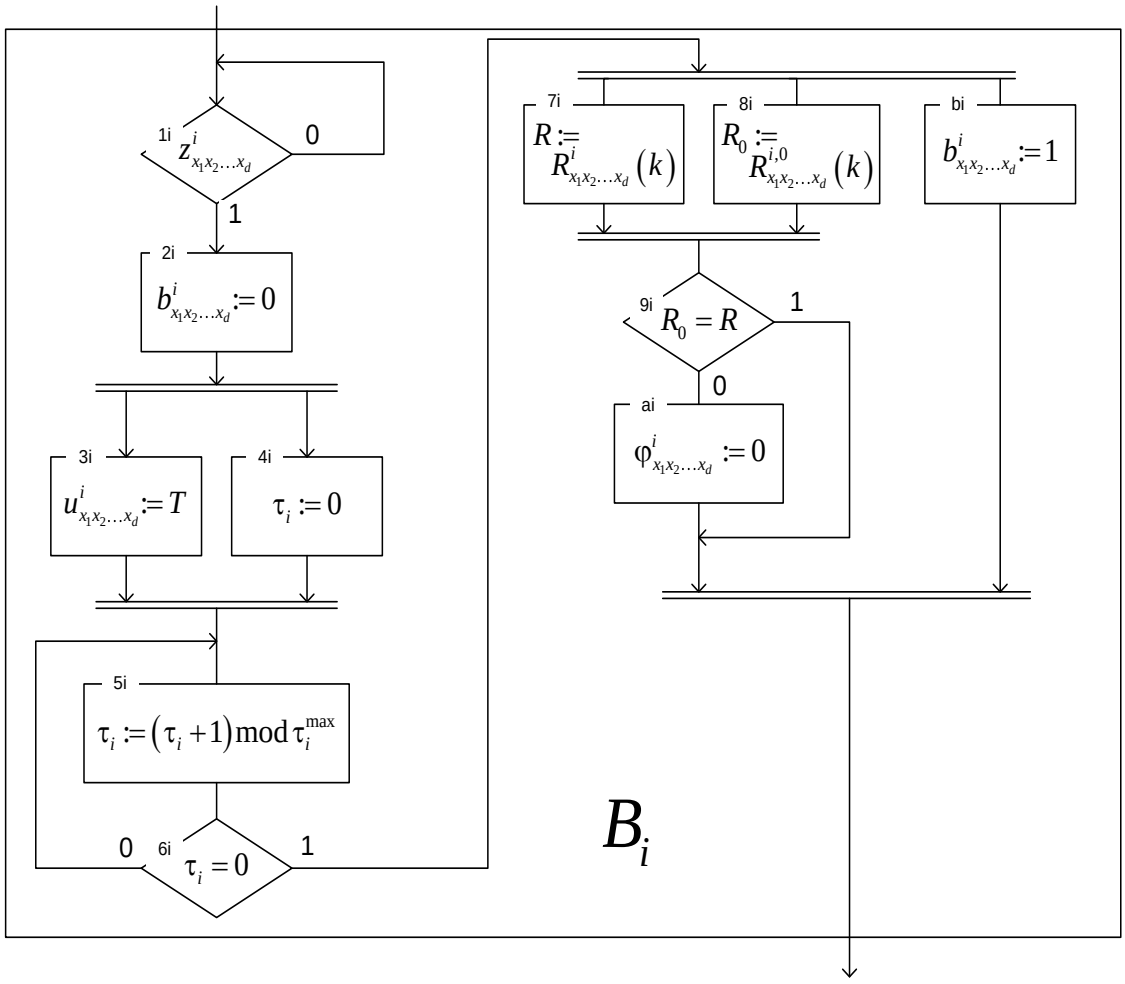


Figure 3. Flow-chart representing thread B_i.

Table 2. Symbols used in the flowchart shown in Figure 3.

No.	Symbol	Meaning
1	$\tau_i^{\max}, i=\overline{0,2d}$	Node $u^i_{x_1x_2\ldots x_d}$ response time limit
2	$\tau_i, i=\overline{0,2d}$	Node $u^i_{x_1x_2\ldots x_d}$ response time counter
3	$R^i_{x_1x_2\ldots x_d}(k)$	Node $u^i_{x_1x_2\ldots x_d}$ response token corresponding to test signature $T^{x_1x_2\ldots x_d}(k)$
4	$R^{i,0}_{x_1x_2\ldots x_d}(k)$	Node $u^i_{x_1x_2\ldots x_d}$ expected response corresponding to test signature $T^{x_1x_2\ldots x_d}(k)$
5	$\varphi^i_{x_1x_2\ldots x_d}$	Node $u^i_{x_1x_2\ldots x_d}$ faulty/healthy decision made by processor $u_{x_1x_2\ldots x_d}$
6	$u^i_{x_1x_2\ldots x_d}$	i^{th} tested node of the current processor
7	R, R_0	Extra buffers

In the first step, thread B_i starts to spin while waiting for condition $z^i_{x_1x_2\ldots x_d}=1$ to become true (see Vertex 1*i*). Immediately, the test (self-test) routine begins for tested neighbor (current node) $u^i_{x_1x_2\ldots x_d}$. Then, flag $b^i_{x_1x_2\ldots x_d}$ is reset (Vertex 2*i*), leading to no other nodes being allowed to test $u^i_{x_1x_2\ldots x_d}$. As a result, no collisions occur, because for any testing unit trying to start checking, the i^{th} tested neighbor $z^i_{x_1x_2\ldots x_d}$ is clear. During the next step, test signature $T^{x_1x_2\ldots x_d}(k)$ is transferred to $u^i_{x_1x_2\ldots x_d}$ (Statement 3*i*) and the timer counts down until $\tau_i^{\max}$ elapses (see Vertices 4*i*–6*i*). As soon as $\tau_i=0$, test response $R^i_{x_1x_2\ldots x_d}(k)$ arrives (or does not in some cases) from tested node $u^i_{x_1x_2\ldots x_d}$ (Statement 7*i*). Concurrently, expected test response $R^{x_1x_2\ldots x_d}(k)$ is fetched (Statement 8*i*) to be compared to $R^i_{x_1x_2\ldots x_d}(k)$ (see Condition 9*i*). If the test response (or whatever has arrived) differs from what is expected to arrive, then $u^i_{x_1x_2\ldots x_d}$ is assumed to be faulty and flag $\varphi^i_{x_1x_2\ldots x_d}$ is reset to zero (Statement a*i*). Otherwise, nothing happens and $\varphi^i_{x_1x_2\ldots x_d}$ remains high. In parallel, flag $b^i_{x_1x_2\ldots x_d}$ is again set high (Statement b*i*), making it possible to self-test or for the other neighbors to test node $u^i_{x_1x_2\ldots x_d}$.

One must mention that all the statements and conditions in the above algorithm are based on simple atomic operations, such as increment, assignment, set/reset, compare and test for zero/one. Therefore, it can be directly implemented in hardware.

5. HARDWARE-LEVEL IMPLEMENTATION

In this section, we discuss the possible hardware-level implementation of the above mutual inter-unit test mechanism. We consider both test units and collision resolution hardware. Taking into account that all processor cores in the multiprocessor are identical, we pick up an arbitrary node for consideration. Assuming a 2-dimensional multiprocessor, we can represent the structure of the test hardware, as shown in Figure 4.

The unit presented in Figure 4 contains 4 identical neighbor check units (NCU1–NCU4) and a self-test unit (STU). The operation of these units is based on the thread algorithm shown in Figure 3. A test organization unit (TOU) is needed to store and fetch test signatures mapped onto the current node in order to control the delay between adjacent test cycles and to coordinate the operation of the NCUs and

STU. TOU implements all the sections of the proposed test algorithm (Figure 2) except for parallel threads $B_0, B_1, \dots, B_{2d}$. Arbitration flip-flops AF0–AF4 organized in a ring shift register are required to perform the proposed round-robin collision resolution scheme to guarantee that two or more neighbors never start checking the same processor within the overlapping time frames. Only one AF can be high at a given moment. This high value moves from one flip-flop to its neighbor and lets each neighboring processor check the current node in a time division manner. A clock pulse generator (CPG) is used to synchronize the operation of the test hardware components. A separate CPG may be employed or the processor's main pulse generator may be considered as a CPG.

The neighbor check units form and issue healthy/faulty flags φ_{xy}^{i} for the corresponding neighbor nodes according to the following rule: $\varphi_{xy}^{i} = 1$ if processor u_{xy} makes a decision that neighbor u_{xy}^{i} is healthy and $\varphi_{xy}^{i} = 0$ otherwise (we assume that $u_{xy}^{0} \equiv u_{xy}$). The same is carried out by the testing neighbors of the current processor. As an addition, node u_{xy} occasionally performs a self-checking routine that results in a healthy/faulty flag φ_{xy}^{0}. Finally, a generalized faulty/healthy flag φ_{xy} is calculated by the majority rule:

$$\varphi_{xy} = \#\left(\varphi_{xy}^{0}, \varphi_{xy}^{1}, \varphi_{xy}^{2}, \varphi_{xy}^{3}, \varphi_{xy}^{4}\right), \tag{7}$$

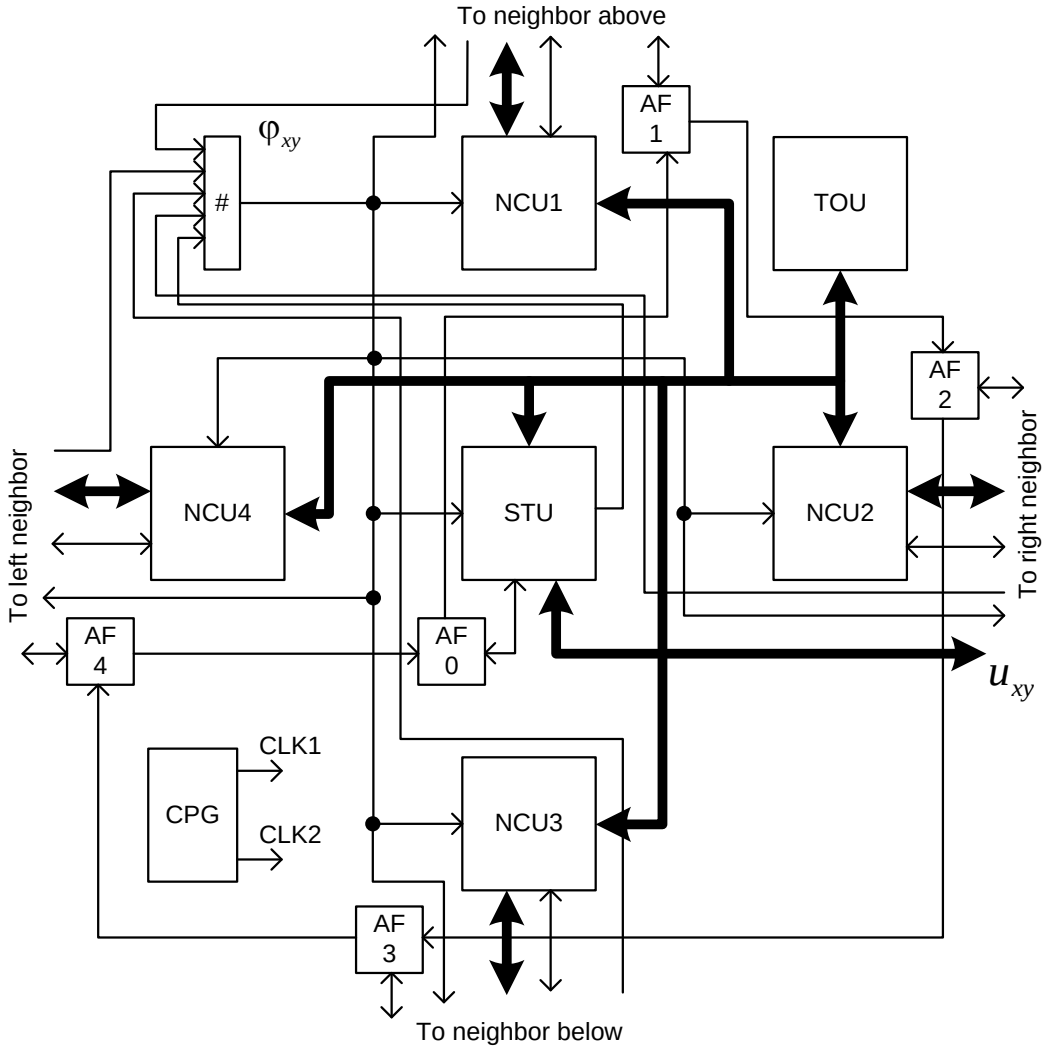


Figure 4. The organization of processor node test hardware in a 2-dimensional multiprocessor.

where # denotes the majority operator. If $\varphi_{xy} = 1$, then u_{xy} is assumed to be healthy. If $\varphi_{xy} = 0$, then it is further treated as faulty.

The structure shown in Figure 4 can be extended to a multiprocessor of any given dimension $d \geq 2$. In a general case, a processor's test hardware includes $2d$ NCUs operating in parallel, $2d$ bidirectional links needed to transfer test signatures to the tested neighbors and to receive response tokens, as well as to receive test signatures from the testing neighbors and transfer response tokens, $2d$ input terminals required to receive healthy/faulty flags from the testing neighbors and $2d$ output terminals to issue the generalized healthy/faulty flag to the direct neighboring processors. The count of the majority gate's input terminals is calculated by Formula (7).

6. THE ROUND-ROBIN ARBITRATION MECHANISM

Using the above conceptual representation, we have developed a functional diagram embodying the round-robin arbitration mechanism necessary to avoid inter-processor test collisions. The scheme of a 2-dimensional multiprocessor is shown in Figure 5, which includes five JK flip-flops with input invertors needed to store the test enable flags. Consequent flip-flop enumeration is adopted and corresponds to the enumeration of the neighbor nodes (the i^{th} flip-flop together with its inverter corresponds to the AFi unit in the block diagram of Figure 4). The flip-flops are connected to each other to form a ring shift register whose operation is clocked by pulse chain CLK1 issued by CPG (not shown in Figure 5 for simplicity). The AND gates are introduced to block pulse chain CLK1 from clocking the flip-flops when the current node is being tested by a neighbor or being self-tested.

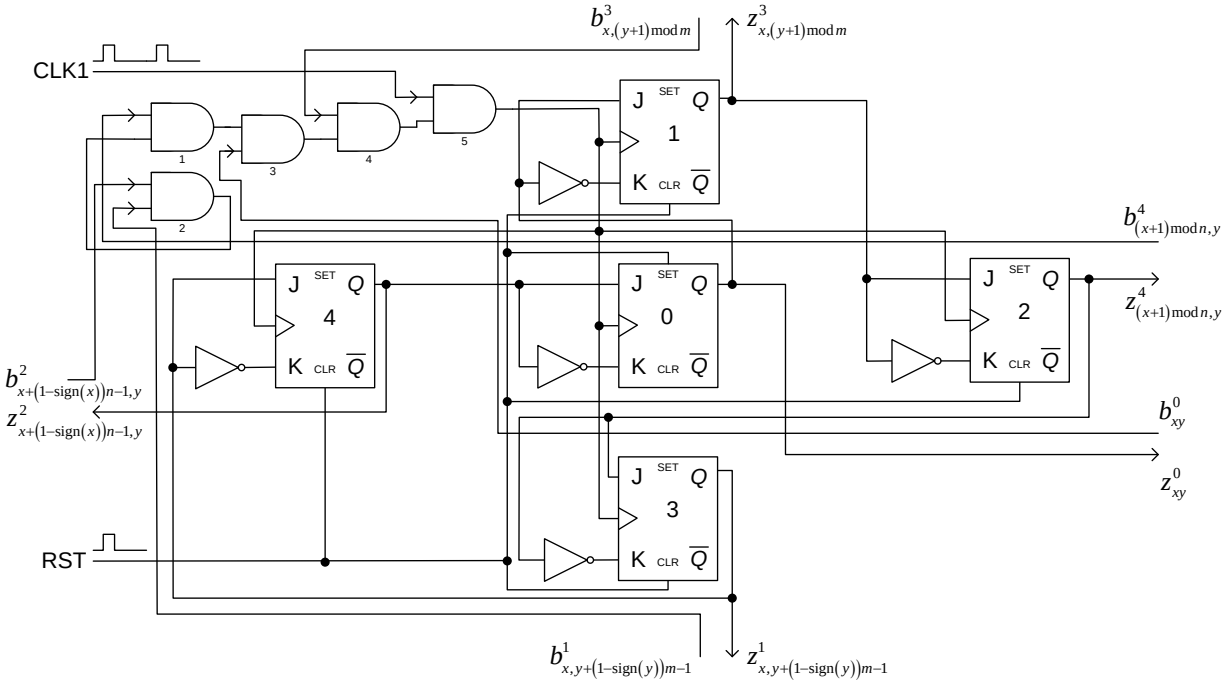


Figure 5. The round-robin arbitration hardware functional diagram for a 2-dimensional multiprocessor.

Initially, a system reset pulse arrives to initialize the flip-flops. As a result, flip-flop 0 is set to "1" while the remaining flip-flops are reset to "0." Thus, the initial state of the test hardware will be in accordance with Statement 2 of the above algorithm (see Figure 2); i.e., all processors are initialized to start self-test routines.

If the current processor is about to start a self-test, then STU (see Figure 4) issues flag $b^0_{xy} = 0$ (which implies that Vertex 2i of thread B_0 will execute). As a result, gate 5 is blocked and no pulse CLK1 is able to clock the flip-flops. Flip-flop 0 remains set while the rest of the flip-flops are reset until the self-test routine terminates. As soon as self-test is done, high flag $b^0_{xy} = 1$ arrives (see Vertex bi of the above algorithm) and the AND gates are unblocked, making it possible to clock the flip-flops. Note that flag b^0_{xy} may remain set and the AND gates may be open if the test routine start timer has not finished counting down yet.

Another pulse CLK1 passes by AND gate 5, feeds the clock inputs of all the flip-flops and transfers the value of logical "1" to flip-flop 1 from flip-flop 0. In turn, flip-flop 0 is reset to zero because of the low

state of flip-flop 4. As a result, flip-flop 1 is set while the remaining peers are reset. After that, flag $Z^3_{x,(y+1)mod\ m} = 1$ arises and allows neighboring processor $u_{x,(y+1)mod\ m}$ to initiate the test process for the current node (superscript "3" implies that current node u_{xy} is the third neighbor of processor $u_{x,(y+1)mod\ m}$).

When processor $u_{x,(y+1)mod\ m}$ starts checking the current node, its NCU3 issues flag $b^3_{x,(y+1)mod\ m} = 0$ (which corresponds to Vertex 2*i* in thread B_3 of the proposed algorithm). AND gate 5 becomes blocked and pulses CLK1 can no longer feed the clock inputs of the flip-flops. Flip-flop 1 stays high while the rest remain clear until the test terminates. As soon as the test is finished, flag $b^3_{x,(y+1)mod\ m} = 1$ arrives (corresponding to Vertex b*i* of our algorithm), the AND gates are unblocked and clock pulses CLK1 start feeding the flip-flops. If flag $b^3_{x,(y+1)mod\ m}$ is not reset to zero (the test routine start timer has not counted down yet), then the AND gates remain open.

Analogously, the high-level value is transferred from flip-flop 1 to flip-flop 2, then travels from flip-flop 2 to flip-flop 3 and finally returns to flip-flop 0 from flip-flop 4, meaning that another arbitration loop is complete. The operation of the test hardware stays the same as discussed above. If flip-flop 2 becomes high, then node $u_{(x+1)mod\ n,y}$ gains the right to start testing the current processor. In turn, when flip-flop 3 is high, neighbor $u_{x,y+(1-sign(y))m-1}$ will check the current processor.

Note that the round-robin arbitration hardware can be easily extended to a *d*-dimensional case, but would include more flip-flops, inverters and terminals. For a 3-dimensional multiprocessor, 7 flip-flops are necessary, whereas 9 flip-flops are required in a 4-dimensional case.

7. COMPARISON OF THE PROPOSED APPROACH

7.1 Probability of Successful Fault Detection Evaluation

To compare the proposed inter-unit test method to the existing alternatives, the probability of successful fault detection is theoretically evaluated first and its dependencies on the multiprocessor dimension and reliability of separate test units are explored. The results are compared to the distributed self-checking and known mutual inter-unit test methods.

We take into account that a faulty node may be erroneously reported as healthy ("hidden fault") by a test unit and that a healthy processor may be mistakenly treated as faulty ("false fault"). All faults that are neither hidden nor false are known as "explicit faults;" i.e., detected faults that really exist. Thus, we define the probability of successful fault detection as a measure of the probability of properly detecting faulty nodes that really exist in the multiprocessor.

Let $\pi(t)$ be the probability that a separate test unit of a processor properly detects a faulty neighbor node (or the current node in the case of self-checking). Let $\pi^-(t)$ and $\pi^0(t)$ be the probabilities that a separate test unit is unable to detect a faulty node and claims a healthy node to be faulty, respectively. Then, the following fundamental relation will take place:

$$\pi(t) = 1 - \left(\pi^-(t) + \pi^0(t)\right). \tag{8}$$

Assuming $\pi(t)$, $\pi^-(t)$ and $\pi^0(t)$ are the same for all the multiprocessor's nodes across the mesh, we deduce the probability of successful fault detection formula for the proposed approach.
For the simplest 2-dimensional case, $|K'_{xy}| = 5$ and we obtain:

$$P(t)\big|_{d=2} = \sum_{i=3}^{5} P_5^i(t) = \sum_{i=3}^{5} C_5^i \pi(t)^i \left[1 - \pi(t)\right]^{5-i}, \tag{9}$$

where $P_5^i(t)$ denotes the probability that *i* out of $|K'_{xy}| = 5$ testing nodes properly detect a faulty neighbor and C_5^i is the number of *i* item selections out of 5 items. For a 3-dimensional multiprocessor, $|K'_{xy}| = 7$ and we have:

$$P(t)\big|_{d=3} = \sum_{i=4}^{7} P_7^i(t) = \sum_{i=4}^{7} C_7^i \pi(t)^i \left[1 - \pi(t)\right]^{7-i}, \tag{10}$$

where $P_7^i(t)$ denotes the probability that i out of $|K'_{xy}| = 7$ testing nodes properly detect a faulty neighbor. Using formulae (9) and (10), we deduce

$$P(t) = \sum_{i=\left\lceil (2d+1)/2 \right\rceil}^{2d+1} P_{2d+1}^i(t) = \sum_{i=\left\lceil (2d+1)/2 \right\rceil}^{2d+1} C_{2d+1}^i \pi(t)^i \left[1-\pi(t)\right]^{2d-i+1}. \quad (11)$$

Using Formula (11), we can investigate the dependencies of the probability of successful fault detection on the multiprocessor dimension and reliability of the separate test units. Having deduced the same formulae for existing approaches, it is possible to compare these approaches to our method on various factors.

Let us first compare our approach to the distributed self-test method. The self-test is able to detect faulty nodes with probability $\pi(t)$, because there is only one test unit in a given processor across the mesh. Therefore, this unit is sufficient to evaluate and explore the relation $\varphi(t) = P(t)/\pi(t)$ in order to compare the proposed method to the distributed self-test. Figure 6 shows the $\varphi(t)$ *versus* probability $\pi(t)$ graphs obtained using Formula (11).

Figure 6 shows that the proposed approach has the greatest advantage when $\pi(t) \approx 0.7$. For a 2-dimensional multiprocessor, the maximum $\varphi(t)$ value attained is 1.1956, which takes place at $\pi(t) = 0.7$ and means that the probability of successful fault detection increases by almost 20% as compared to the distributed self-test approach. The higher the multiprocessor dimension *d*, the greater the advantage of our method. For example, in a 5-dimensional mesh-connected multiprocessor, $\varphi(t)$ becomes higher than 1.3 at $\pi(t) = 0.7$, which signifies a 30% advantage. For higher values $\pi(t) \geq 0.9$, our method becomes less advantageous than a simple self-test. However, higher reliability test units are hard to build up in practice. For lower values $\pi(t) \leq 0.6$, our method also works worse and for $\pi(t) \leq 0.5$, it does not work at all. However, the case $\pi(t) \leq 0.5$ corresponds to "extremely unreliable" test units, which (according to Formula (8)) would claim faulty units to be healthy and/or treat healthy units as faulty in most cases such that a self-test would not be feasible with such units.

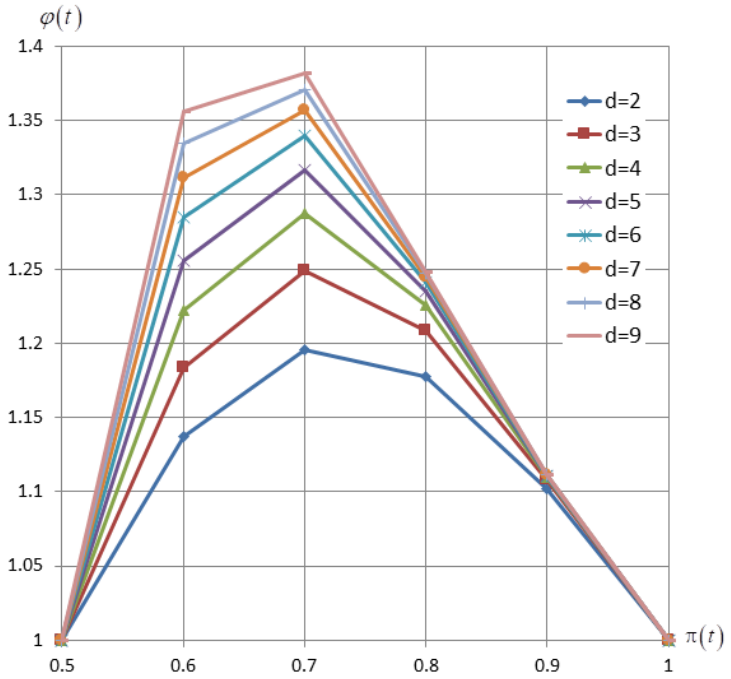


Figure 6. $\varphi(t)$ *versus* $\pi(t)$ graphs for fixed $2 \leq d \leq 9$.

Let us now compare our approach to the mutual inter-unit test method presented in [41] under the same assumptions as those formulated above. Let $P_0(t)$ denote the probability of fault detection attained when the mutual inter-unit test [41] is employed. Then, it is sufficient to evaluate and explore the relation $\psi(t) = P(t)/P_0(t)$ to compare the proposed method to the mutual inter-unit test. Figure 7 presents the $\psi(t)$ *versus* probability $\pi(t)$ graphs obtained using Formula (11) and a similar formula found in [41].

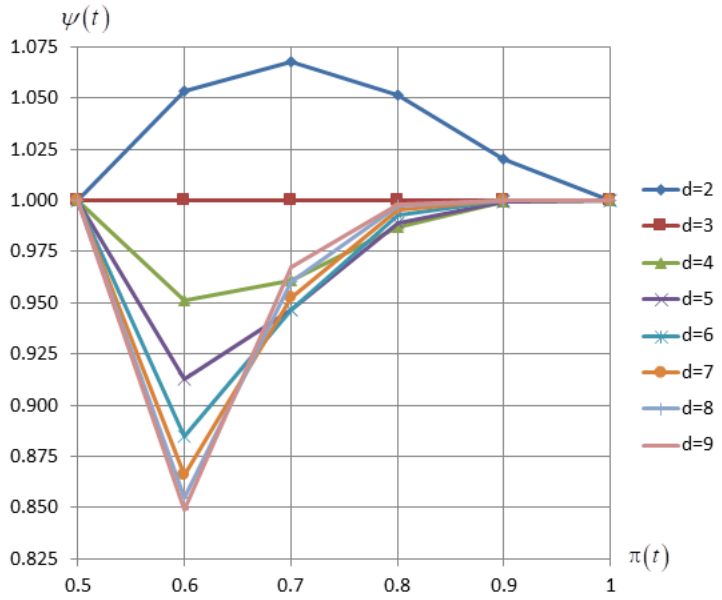


Figure 7. $\psi(t)$*versus* $\pi(t)$ graphs for fixed $2 \le d \le 9$.

Analyzing the graphs of Figure 7, we can see that our approach works better for 2-dimensional multiprocessors. It is evident that these results are from the higher testing neighbor set cardinality. With $d = 2$ and $\pi(t) = 0.7$, the advantage of our approach is about 7%. In the 3-dimensional case, our method demonstrates the same probability $P(t)$ values as does the inter-unit test. For $d \ge 4$, our method becomes slightly worse than the inter-unit test at $\pi(t) \ge 0.7$ and loses at $\pi(t) \approx 0.6$, for which the probability is actually a bit too low for practical cases.

7.2 Connection Complexity Evaluation

Excessive connectivity is the main drawback of known mutual inter-unit test approaches. To deploy an inter-unit test environment, each processor node needs many external connections (input and output terminals) to communicate to its peers while performing test routines. This connectivity depends highly on the multiprocessor dimension and test unit parameters and so, may become a serious concern when complex systems are being manufactured.

In what follows below, we compare the connectivity factor in our approach to those in the known mutual inter-unit test methods and demonstrate that our proposed method can drastically decrease multiprocessor connectivity. The connection complexity is formally defined as the required number of extra direct connections between a given processor and all its peers to perform test routines. Only extra test connections are considered. It is assumed that the "regular" connections required for inter-processor data exchange and control are the same whichever test method is used, but only external connections are under consideration. For example, the links between the processor core and STU are not taken into account, because they are internal.

According to Figs. 4 and 5, i^{th} NCU needs $\Omega_i = W_R + W_T + 4$ input/output terminals, where W_R and W_T are the widths of response packets and test signatures, respectively. Thus, taking into account the round-robin arbitration unit connections, the majority gate terminals and the backward test/response buses, the total number of extra input/output terminals of a processor may be calculated as:

$$\Omega = 2d\left(2\left(W_R + W_T + 4\right) + 1\right). \tag{12}$$

In the same fashion, the connection complexity of known mutual inter-unit test schemes can be evaluated. For the inter-unit test method presented in [41], the following formula will take place:

$$\Omega_0 = 2\left[\left(d\left(d-1\right)+1\right)\left(W_R + W_T + 1\right) + d\right]. \tag{13}$$

Formula (13) takes into account all extra test connections to the peers of a given processor.

To compare our approach to the inter-unit test method of [41], we calculate the relation $\xi = \Omega_0/\Omega$ for different values of the multiprocessor dimension *d* and fixed W_R and W_T (We assume that $W_R = W_T$ for simplicity reasons). In Figure 8, ξ *versus* d graphs are shown for $W_R + W_T \in \{32,64,128,256\}$. According to the graphs of Figure 8, our approach requires a few more extra connections in the 2-dimensional case. For $d > 2$, our method works better than that of [41]. For example, given a 4-dimensional multiprocessor, the inter-unit test method would require 55% more extra connections than does our approach. Thus, with our proposed test method, higher dimension fault-tolerant multiprocessors would be significantly easier to implement because of the lower extra connectivity.

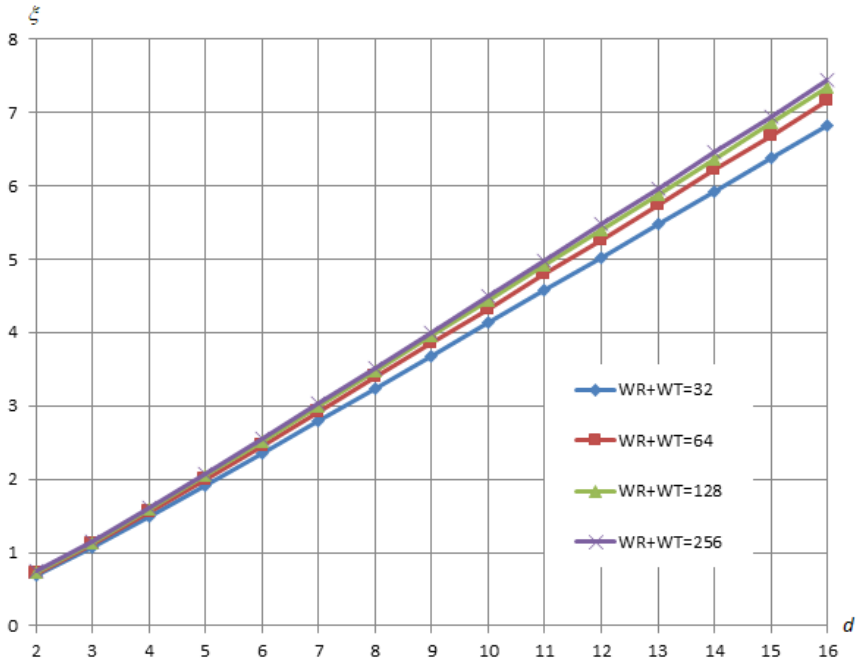


Figure 8. ξ *versus* *d* graphs for fixed $W_R + W_T \in \{32,64,128,256\}$.

8. CONCLUSION

In this paper, we proposed a new approach of a mutual inter-unit test with round-robin collision resolution to improve the testability of mesh-connected multiprocessors by increasing the probability of successful fault detection as compared with simple distributed self-checking. Compared with other mutual inter-unit test methods, such as [41] and [42], our approach automatically resolves the collision problem when two or more neighboring processors are about to start checking the same peer during overlapping time windows. Our method can be applicable to multiprocessors of arbitrary dimensions, with 2-dimensional ones having the maximum effectiveness, which matches the technological limitations of modern VLSI multiprocessors. For future scaling, our approach must allow drastic reduction of the multiprocessor connectivity with respect to known mutual inter-unit test methods. For example, in a 4-dimensional system, we need 55% less extra connections with our approach, while in a 5-dimensional case, we reduce extra connectivity by over 90%. The new mutual inter-unit test technique allows for the online hardware-level testing of all processor nodes across the mesh in parallel, thereby significantly contributing to the performance of the test environment.

REFERENCES

[1] M. Abramovici, M. A. Breuer and A. D. Friedman, "Digital Systems Testing and Testable Design," IEEE Press, Piscataway, NJ, 1994.

[2] M.K. Aguilera, W. Chen and S. Toueg, "Failure Detection and Consensus in the Crash-Recovery Model," Distributed Computing, vol. 13, no. 2, pp. 99–125, 2000.

[3] R. Ahlswede and H. Aydinian, "On Diagnosability of Large Multiprocessor Networks," Discrete Applied Mathematics, vol. 156, no. 18, pp. 3464–3474, Nov. 2008.

[4] L. Benini and G. De Micheli, "Networks on Chips: A Paradigm," IEEE Transactions on Computers,

vol. 35, no. 1, pp. 70–78, 2002.

[5] P. Bernardi, L.M. Ciganda, E. Sanchez and M. Sonza Reorda, "MIHST: A Hardware Technique for Embedded Microprocessor Functional On-Line Self-Test," IEEE Transactions on Computers, vol. 63, no. 11, pp. 2760–2771, Nov. 2014.

[6] R. Bianchini and R. Buskens, "Implementation of On-Line Distributed System-Level Diagnosis Theory," IEEE Transactions on Computers, vol. 41, pp. 616–626, May 1992.

[7] T. Bjerregaard and S. Mahadevan, "A Survey of Research and Practices of Network-on-Chip," ACM Computing Surveys, vol. 38, no. 1. pp. 1–51, 2006.

[8] D. Blough and H. Brown, "The Broadcast Comparison Model for On-Line Fault Diagnosis in Multicomputer Systems: Theory and Implementation," IEEE Transactions on Computers, vol. 48, pp. 470–493, May 1999.

[9] B. Ciciani, Ed., Manufacturing Yield Evaluation of VLSI/WSI Systems, Los Alamitos, CA: IEEE Computer Society Press, 1998.

[10] S. R. Das, "Self-testing of Cores-based Embedded Systems with Built-in Hardware," IEE Proceedings–Circuits, Devices and Systems, vol. 152, no. 5, pp. 539–546, Oct. 2005.

[11] E. P. Duarte Jr. and T. Nanya, "A Hierarchical Adaptive Distributed System-Level Diagnosis Algorithm," IEEE Transactions on Computers, vol. 47, pp. 34–45, Jan. 1998.

[12] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw and D. Sylvester, "Vicis: A Reliable Network for Unreliable Silicon," Proc. of the 46th DAC, pp. 812–817, Jul. 2009.

[13] S. Furber, "Living with Failure: Lessons from Nature?," Proc. of the 11th IEEE ETS, pp. 4–8, May 2006.

[14] T. Horita and I. Takanami, "Fault-tolerant Processor Arrays based on the 1.5-track Switches with Flexible Spare Distributions," IEEE Transactions on Computers, vol. 49, no. 6, pp. 542–552, June 2000.

[15] S. Y. Hsieh and C. Y. Kao, "The Conditional Diagnosability of k-Ary n-Cubes under the Comparison Diagnosis Model," IEEE Transactions on Computers, vol. 62, no. 4, pp. 839 – 843, April 2013.

[16] L. M. Huisman, "Diagnosing Arbitrary Defects in Logic Designs Using Single Location at a Time (SLAT)," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, no. 1, pp. 91–101, 2004.

[17] S. M. A. H. Jafri, S. J. Piestrak, O. Sentieys and S. Pillement, "Design of the Coarse-grained Reconfigurable Architecture DART with On-line Error Detection," Microprocessors and Microsystems, vol. 38, no. 2, pp. 124–136, 2014.

[18] G. Jiang, W. Jigang and J. Sun, "Efficient Reconfiguration Algorithm for Three-dimensional VLSI Arrays," Proc. of the IEEE 26th International Parallel and Distributed Processing Symposium Workshops & Ph.D. Forum, pp. 261–265, 2012.

[19] W. Jigang, T. Srikanthan, G. Jiang and K. Wang, "Constructing Sub-Arrays with Short Interconnects from Degradable VLSI Arrays," IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 4, pp. 929–938, April 2014.

[20] A. Kohler and M. Radetzki, "Fault-tolerant Architecture and Deflection Routing for Degradable NoC Switches," Proc. of the 3rd ACM/IEEE Int. Symp. NoC, pp. 22–31, May 2009.

[21] E. Kolonis, M. Nicolaidis, D. Gizopoulos, M. Psarakis, J. Collet and P. Zajac, "Enhanced Self-configurability and Yield in Multicore Grids," Proc. of the 15th IEEE IOLTS, pp. 75–80, Jun. 2009.

[22] A. Krstic, W. C. Lai, K. T. Cheng, L. Chen and S. Dey, "Embedded Software-based Self-test for Programmable Core-based Designs," IEEE Design and Test of Computers, vol. 19, no. 4, pp. 18–27, July/Aug. 2002.

[23] J. C. M. Li and E. J. McCluskey, "Diagnosis of Resistive-Open and Stuck-Open Defects in Digital CMOS Ics," Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 11, pp. 1748–1759, 2005.

[24] L. Lin, S. Zhou, L. Xu and D. Wang, "The Extra Connectivity and Conditional Diagnosability of Alternating Group Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 8, pp. 2352–2362, Aug. 2015.

[25] S. Lin, W. Shen, C. Hsu, C. Chao and A. Wu, "Fault-tolerant Router with Built-in Self-test/Self-diagnosis

and Fault-isolation Circuits for 2D Mesh-based Chip Multiprocessor Systems," Proc. Int. Symp. VLSI-DAT, pp. 72–75, Apr. 2009.

[26] P. Maestrini and P. Santi, "Self-diagnosis of Processor Arrays Using a Comparison Model," Proc. Of the 14th Symp. on Reliable Distributed Systems, pp. 218–228, 1995.

[27] J. Mekkoth, M. Krishna, J. Qian, W. Hsu, C.-H. Chen, Y. S. Chen, N. Tamarapalli, W. T. Cheng, J. Tofte and M. Keim, "Yield Learning with Layout-Aware Advanced Scan Diagnosis," Proc. of the International Symposium for Testing and Failure Analysis, pp. 412–418, 2006.

[28] M. Psarakis, D. Gizopoulos, E. Sanchez and M. Sonza Reorda, "Microprocessor Software-based Self-testing," IEEE Design and Test of Computers, vol. 27, no. 3, pp. 4–19, May/June 2010.

[29] J. Raik and V. Govind, "Low-area Boundary BIST Architecture for Meshlike Network-on-Chip," Proc. of the 15th IEEE Int'l Symp. DDECS, pp. 95–100, Apr. 2012.

[30] J. Rajski, J. Tyszer, M. Kassab and N. Mukherjee, "Embedded Deterministic Test," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, no. 5, pp. 776–792, 2004.

[31] S. Rangarajan, A. T. Dahbura and E. Ziegler, "A Distributed System-Level Diagnosis Algorithm for Arbitrary Network Topologies," IEEE Transactions on Computers, vol. 44, pp. 312–334, Feb. 1995.

[32] S. Rangarajan and D. Fussell, "Diagnosing Arbitrarily Connected Parallel Computers with High Probability," IEEE Transactions on Computers, vol. 41, pp. 606–615, May 1992.

[33] A. Sengupta and A. T. Dahbura, "On Self-Diagnosable Multiprocessor Systems: Diagnosis by the Comparison Approach," IEEE Transactions on Computers, vol. 41, pp. 1386–1396, Nov. 1992.

[34] M. Sharma, C. Schuermyer and B. Benware, "Determination of Dominant-Yield-Loss Mechanism with Volume Diagnosis," Proc. of IEEE Design & Test of Computers, vol. 27, no. 3, pp. 54–61, 2010.

[35] C. Stroud, J. Sunwoo, S. Garimella and J. Harris, "Built-in Self-test for System-on-Chip: A Case Study," Proc. of the Int'l Test Conf., pp. 837–846, 2004.

[36] W. C. Tam, O. Poku and R. D. Blanton, "Systematic Defect Identification through Layout Snippet Clustering," Proc. of the IEEE International Test Conference, pp.1, 2010.

[37] H. Tang, S. Manish, J. Rajski, M. Keim and B. Benware, "Analyzing Volume Diagnosis Results with Statistical Learning for Yield Improvement," Proc. of the European Test Symp., pp. 145–150, 2007.

[38] Z. Wang, M. Marek-Sadowska, K. H. Tsai and J. Rajski, "Analysis and Methodology for Multiple-Fault Diagnosis," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 25, no. 3, pp. 558–575, 2006.

[39] L. Zhang, "Fault-Tolerant Meshes with Small Degree," IEEE Transactions on Computers, vol. 51, no. 5, pp. 553–560, May 2002.

[40] Z. Zhang, D. Refauvelet, A. Greiner, M. Benabdenbi and F. Pecheux, "On-the-Field Test and Configuration Infrastructure for 2-D-Mesh NoCs in Shared-Memory Many-Core Architectures," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 22, no. 6, pp. 1364–1376, June 2014.

[41] J. Al-Azzeh, M. E. Leonov, D. E Skopin, E. A. Titenko and I. V. Zotov, "The Organization of Built-in Hardware-Level Mutual Self-Test in Mesh-Connected VLSI Multiprocessors," International Journal on Information Technology, vol. 3, no. 2, pp. 29–33, 2015.

[42] J. Al-Azzeh, "A Distributed Multiplexed Mutual Inter-Unit in-Operation Test Method for Mesh-Connected VLSI Multiprocessors," Jordan Journal of Electrical Engineering, vol. 3, no. 3, pp. 193-207, 2017.

ملخص البحث:

في هذه الورقة، يتم اقتراح طريقة جديدة للفحص المتبادل بين الوحدات في المعالجات الدقيقة متعددة الأبعاد في نطاق تقنية التكامل واسع النطاق جدّاً، وذلك لضمان أنّ أي معالج يجري فحصه فقط من قبل العُقدة المجاورة له دون غيرها في الوقت ذاته وأنّه ليست هناك حاجة لبذل عناية خاصة لاختيار اللحظات التي يجب أن تبدأ فيها عمليات الفحص؛ بمعنى أن الطريقة المقترحة تحول دون وقوع تعارض في فحص الوحدات. وقد تم بناء خوارزمية خاصة لهذا الغرض.

وقد أثبتت الطريقة المقترحة أنها تنطوي على تحسين لفحص المعالجات الدقيقة المتصلة في هيئة شبكة؛ من خلال زيادة الاحتمالية للكشف الناجح عن الأخطاء مقارنة بطريقة الفحص الذاتي الموزعة. من جانب آخر، تقلل الطريقة المقترحة بشكل حادّ من الوصلات اللازمة في المعالج الدقيق إذا قورنت بالطرق الأخرى المعروفة للفحص المتبادل بين الوحدات، ومن ثم فهي تسهل من عملية تصنيع المعالجات الدقيقة. فعلى سبيل المثال، في نظام رباعي الأبعاد، تقود الطريقة المقترحة الى تخفيض الوصلات الإضافية اللازمة بنسبة 55%.

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 05, No. 01, April 2019.

MODIFIED RANDOM BIT CLIMBING (λ -MRBC) FOR TASK MAPPING AND SCHEDULING IN WIRELESS SENSOR NETWORKS

Yousef E. M. Hamouda

(Received: 8-Nov.-2018, Revised: 17-Dec.-2018, Accepted: 23-Dec.-2018)

ABSTRACT

This paper examines the problem of Task Mapping and Scheduling (TMS) in Wireless Sensor Networks (WSNs). The application, which is supposed to be executed in WSNs, can be divided into interdependent tasks. The key objectives of TMS in WSNs are the improvement of execution time, energy consumption and network lifetime. A modified version of Random Bit Climbing (RBC) optimization method, also called λ-Modified Random Bit Climbing (λ-mRBC), is developed to get better and faster optimal or near-optimal solution. In the proposed λ-mRBC method, a new operator, called transposition operator, is added to improve the exploration of search space and hence to escape from the local optima. The deepth of exploration is controlled by using a single parameter (λ). Firstly, a number of sensor nodes is selected to cooperatively execute the application with the purpose of improving the network lifetime. After that, the proposed λ-mRBC method is performed to get the optimal or near-optimal task/sensor pair solution, so that the execution time and energy consumption are minimized.

The simulation results show that λ-mRBC method enhances the TMS performance. Compared with the traditional RBC method, the proposed λ-mRBC method converges to better fitness value, make-span and total energy consumption by 19.1%, 19.6% and 22.3%, respectively. Furthermore, the network lifetime is prolonged through using the proposed selection algorithm. The distribution of remaining energy among sensor nodes is improved about three times, compared with the random selection scheme. Furthermore, compared with the random selection, the number of neighbours for sensor nodes is improved by 20.1% using the proposed selection algorithm.

KEYWORDS

Application DAG, Optimization methods, Random bit climbing, Task mapping and scheduling, Wireless sensor networks.

1. INTRODUCTION

Sophisticated technologies and applications, such as smart homes, Internet of Things (IoT), smart grid, precision agriculture and automated control have provoked the need of developing self-organized, multi-hop and *ad-hoc* Wireless Sensor Networks (WSNs). WSNs are made up of hundreds or thousands of tiny and cheap sensor nodes with limited resources. Sensor nodes cooperate with each other to execute the applications. In addition, sensor nodes are scattered randomly or in a planned manner to monitor and control the field of interest [1]. Sensor node consists of energy unit, processing unit, sensing unit, wireless communication unit and storage unit [2]. In several applications, WSNs are positioned in sites that are difficult to be physically accessed; i.e., forest. Therefore, network lifetime is an essential requirement for WSNs to prolong the lifetime of the sensor nodes and the network connectively [3, 4]. Lots of civil and military applications employ WSNs. Civil applications, for example, include healthcare [5], precision irrigation [6], smart grid [7], home automation [8] and surveillance [9], while military-based applications usually include intrusion detection and detection of illegal crossings [10].

Given the fact that the sensor nodes have limited resources, improving the energy-efficiency and application execution time of WSNs seems to be plausible to increase the network lifetime [11]. In fact, energy is consumed from the battery during sensing, communicating and processing activities. In

Y. E. M. Hamouda is with Department of Computer, Al-Aqsa University, Gaza, Palestine. Email: ye.hamouda@alaqsa.edu.ps

addition, numerous applications in WSNs require massive in-network processing capability. For instance, smart visual sensor networks usually go through several subsequent executional jobs, including image processing, computer vision and image sensing [12]-[13]. Furthermore, application should complete its execution at the right time, after which the execution of the application will not be useful anymore. In most cases, sensor node is not fast enough to execute the complex application in a reasonable amount of time.

Parallel computing refers to the methods that solve a problem within a reasonable amount of time (i.e., called make-span) by dividing it into smaller parts and solving the parts using multiple physical processors [14]. As a result, a complicated application of WSNs is decomposed into smaller tasks. Afterward, Task Mapping and Scheduling (TMS) techniques are developed in order to share the execution of divided tasks among sensor nodes [15]. As shown in Figure 1, four phases are required to perform a parallel system [35]: The first phase is called task decomposition. In this phase, the application is divided into small tasks. These small tasks depend on each other by using dependencies. The dependence analysis is performed in the second phase to order the tasks in line with the dependencies. Task graphs, such as Direct Acyclic Graphs (DAGs), are employed to model the tasks with dependencies. The third phase is called task mapping and scheduling. The main purpose of task mapping is to allocate the tasks to the processors or computing nodes. Consequently, the task scheduling aims to order the task execution according to the dependencies through determining the start times of task execution. The fourth phase is called parallel programming and aims to develop the application, based on the result of task mapping and scheduling. This paper considers the mechanism of task mapping and scheduling, explained in the third phase.

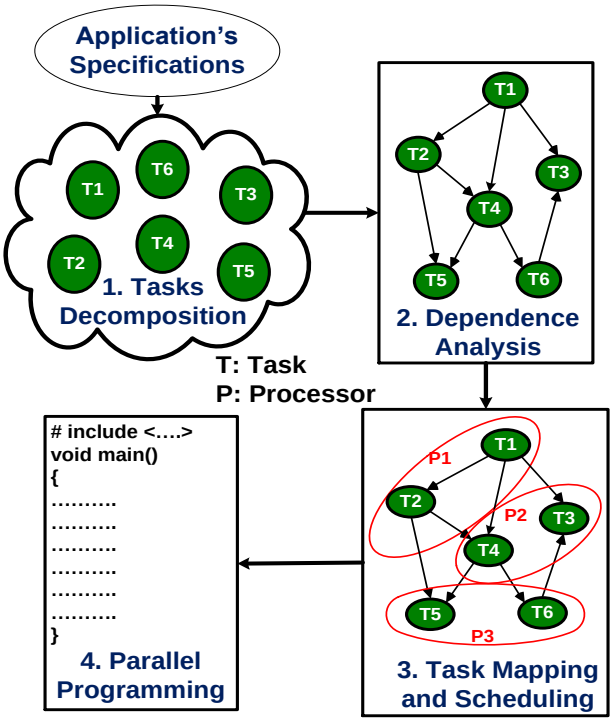


Figure 1. Task mapping and scheduling concept.

However, unlike the traditional parallel computing systems, TMS in WSNs focuses not only on the execution time of the application, but also on energy-efficient schemes that prolong the network lifetime. Therefore, TMS is generally modelled as a multi-objective optimization problem, which has been proved to be non-deterministic polynomial-time (NP)-hard [16].

Metaheuristics are used to get a satisfactory solution of optimization problem that fulfils the required objective function. Metaheuristics are strategies that depend on a guide to examine the search space to get the optimal (or near-optimal) solution, without the need of testing every solution in the search space [17]. A metaheuristic has two main properties: diversification and intensification [18]. Diversification is the exploration of search space in order to escape from the local optimal. Intensification refers to the process of exploitation of accumulated search space. Indeed,

diversification needs a time to be performed, whereas intensification looks deeply and locally for high-quality solutions. Therefore, dynamic balancing between exploration and exploitation is necessary for a good metaheuristic [17]. In literature, metaheuristics are classified into single-solution metaheuristics and population-based metaheuristics. According to the operation form of metaheuristics, single-solution metaheuristics are of more intensification, while population-based metaheuristics are of more diversification [18].

This paper introduces λ-Modified Random Bit Climbing (λ-mRBC) optimization method to attempt to solve the problem of TMS in WSNs. The claim of this research is to improve the traditional RBC optimization method. The main contributions of the proposed λ-mRBC optimization method are: (1) a novel modification in RBC method is developed to improve the convergence speed and fitness value of the final solution. (2) The researcher thinks that this research might be the first one to apply the RBC method and its proposed modified version (λ-mRBC) in TMS problem. (3) Lifetime Awareness Sensor Node Selection Algorithm (LA-SNSA) is incorporated to select the sensor nodes so that the lifetime of the network is improved. (4) Heterogeneous sensor nodes with different processing, energy level and energy consumption are used in the proposed algorithms.

The paper contains seven sections as follows: Section (2) explores work related to task allocation in WSNs. In Section (3), the network framework for TMS is introduced. Then, Section (4) defines and formalizes the research problems. After that, the proposed λ-mRBC and LA-SNSA schemes are introduced in Section (5). Simulation results are shown and discussed in Section (6). Finally, the paper is concluded in Section (7).

2. RELATED WORK

Task mapping and scheduling problems have been deeply discussed in WSNs. In [19]-[20], Genetic Algorithm (GA) is used to provide well-performing task allocation. A Modified Binary Particle Swarm Optimization (MBPSO) algorithm is presented in [21] to find the optimal task allocation solution. In [22], logic gate-based evolutionary algorithm is used to solve the problem of task allocation in WSNs. However, the population-based metaheuristics used in the above research require high processing power, energy consumption and execution time. Furthermore, high complexity optimization algorithms are not appropriate for limited resource WSNs.

Integer linear programming is used in [23] to optimally assign complex tasks to sensor nodes to minimize total energy consumption. In [24], task allocation is introduced so that energy consumption and network lifetime are improved. However, the execution time (i.e., make-span) has not been taken into account [23]-[24], which leads the application to take long time to be executed.

In [25], a distributed task allocation is introduced. The task is made to move from a sensor node to another. The suitable sensor node with enough capacity to execute the task is found. In [26], Topology-Aware Task Allocation and Scheduling (TATAS) is introduced to map and schedule the tasks to the sensor nodes. However, the task allocation presented in [25]-[26] assumes independent tasks which are not practical for complex application, such as visual surveillance [27].

In [28], a real-time task mapping and scheduling (RT-MapS) algorithm is developed for collaborative in-network processing in single-hop cluster WSN using Dynamic Voltage Scaling (DVS) feature. In [27], Multi-hop Task Mapping and Scheduling (MTMS) solution is developed for TMS in multi-hop cluster WSN. Nevertheless, MTMS and RT-MapS prevent task mapping to sensor nodes that execute the immediate predecessors of the task. As a result, this leads to using more sensor nodes for TMS and including all sensor nodes in the task mapping decision-making.

In [29], Biological Task Mapping and Scheduling (BTMS) approach is introduced, where the application is executed by a group of sensor nodes so that the execution time and energy consumption are improved. However, the network lifetime related to sensor neighbour count is not considered. In [30], Light Allocation of Tasks (LAT) algorithm is presented to enhance energy efficiency, network lifetime and application execution time. However, LAT algorithm includes all sensor nodes in decision-making for TMS.

Task Level Parallelism (TLP) in WSN is introduced in [31] to parallelize the execution of smart health care applications so that the processing time is reduced. Nevertheless, scheduling of the task execution

is not considered. An energy-efficient Complicated Task Solution scheme for real-time task processing based on node Cooperation (CTSC) is tackled in [32] to allocate more tasks to sensor nodes with a higher energy-level. However, CTSC maps all dependent tasks to the same sensor nodes which could cause exhaustion for the energy level of sensor node.

In [33], Machine-to-Machine (M2M) architecture with sensor devices and limited resources is considered. Tasks are allocated to the nodes of M2N so that the lifetime is maximized. However, the task allocation algorithm proposed in [33] finds all possible task allocation possibilities which need high processing and time. In addition, the execution time is not considered in [33]. In [34], complex application is allocated for different clustered wireless sensors. Firstly, tasks are distributed to clusters so that the energy consumption is minimized. Then, tasks allocated to each cluster are assigned to the nodes within the cluster so that energy cost and load balancing are improved. In [35], the problem of task allocation in IoT applications is considered, where the embedded devices of IoT are assumed to have limited resources. The tasks are allocated so that the energy consumption is minimized. In [36], tasks are allocated locally to the slave sensor nodes or globally to the master sensor node, so that the network life time is maximized. However, the execution time is not considered in [34]-[36].

In this paper, λ-Modified Random Bit Climbing (λ-mRBC) optimization method is developed to solve the problem of TMS in WSNs. The proposed method supports heterogeneous sensor nodes. Actually, the proposed λ-mRBC method is different from the previous one through the use of a modified version of RBC method with faster conversion speed and better final solution. Moreover, the proposed λ-mRBC is a single-solution metaheuristic with single algorithm parameter. Therefore, it needs less processing capabilities to be executed and in turn it is suitable for the sensor nodes with limited resources. Finally, Lifetime Awareness Sensor Node Selection Algorithm (LA-SNSA) is developed to select the sensor nodes to enhance network lifetime.

3. NETWORK FRAMEWORK

As shown in Figure 2, the sensor nodes are randomly distributed in the monitoring area. The sensor nodes are connected with each other wirelessly. The sink node aims to pass on the data from the monitoring area to the main controller *via* Internet, satellite or cellular networks. Sensor node knows its location using Global Positioning System (GPS) [37]. Nonetheless, only few sensor nodes use GPS to know their locations and other sensor nodes can calculate their locations using triangulation [38]. At time step (k), the neighbours of a target sensor node (s_{TSN}) are a set $N_{s_{TSN}}(k)$. The neighbours with remaining energy level above a predefined threshold value (E_{th}) can participate to execute an application DAG (A_d). These particular neighbours are saved in an $S_s(k, A_d)$ set of $n_s(k, A_d)$ sensor nodes. After that, a set $(S_g(k, A_d))$ of $n_g(k, A_d)$ nodes is selected from $S_s(k, A_d)$ to execute the application DAG (A_d). The selection of $n_g(k, A_d)$ sensor nodes is performed to improve the network lifetime. The application DAG (A_d) is assumed to be decomposed into interdependent tasks. Then, TMS is incorporated to cooperatively execute the application tasks using the selected sensor nodes, so that the time and energy required to execute the application DAG are reduced.

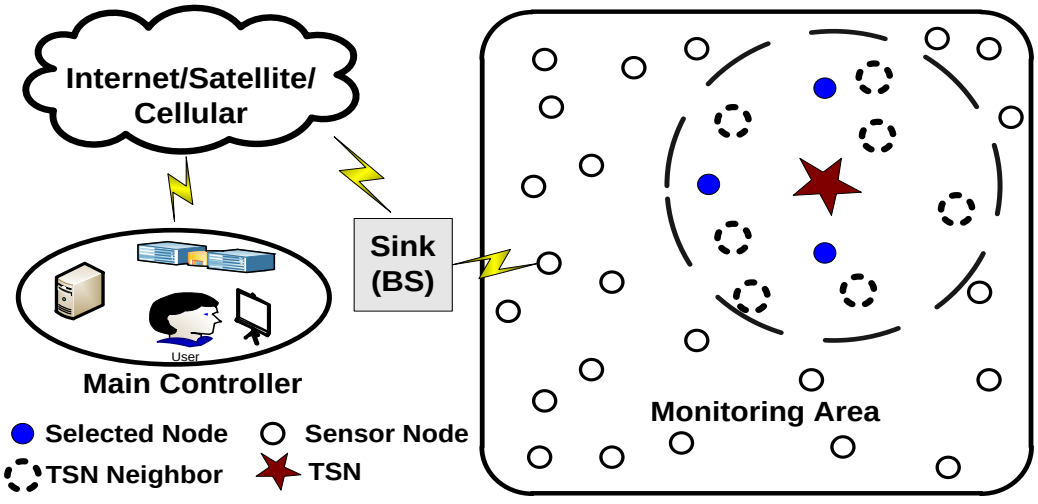


Figure 2. Network framework.

4. PROBLEM DEFINITION AND FORMALIZATION

4.1 Application Model

In this paper, the application is modelled using Direct Acyclic Graphs (DAG) [21]. So, the application is divided into smaller tasks. DAG can also model the interdependencies among tasks [39]. Figure 3 shows as example of application DAG.

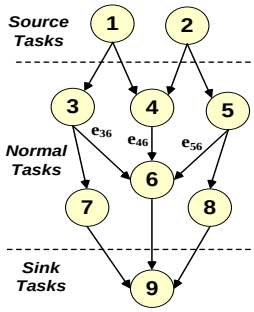


Figure 3. Application DAG.

The application DAG is modelled as $A_d = (V, E)$. The set V represents "n" application tasks, where $V = \{v_i : i = 1,2,..n\}$. Similarly, the set E represents "q" communication interdependencies, where $E = \{e_k : k = 1,2,..q\}$. The edge $e_k \in E$ between the tasks v_i and v_j is denoted as e_{ij}, where v_j is called the immediate successor of v_i and v_j is called the immediate predecessor of v_i. Accordingly, the task is executed when it receives all of its immediate predecessor's output. The entry-tasks or source-task do not have immediate predecessors. In addition, a task without immediate successors is called an exit-task or a sink-task. In WSNs, the entry-tasks are used for sensing or gathering the raw data to detect physical phenomena. Therefore, task placement constraints can be defined as an only one source task that can be assigned to the sensor node. In Figure 3, v_1 and v_2 are source-tasks, v_9 is the sink-task, v_1 and v_2 are the immediate predecessors of v_4 and v_9 is the immediate successor of v_8. The task v_6 cannot be executed until it receives the communication interdependencies (e36, e46 and e56) from its immediate predecessors (v_3 , v_4 and v_5).

Each task, $v_i \in V$ is modelled as a tuple of the form: $\{N_{v_i}, t_{v_i}, E_{v_i}\}$, where N_{v_i} is the number of the computational cycles of the task, E_{v_i} is computational energy consumption of the task and t_{v_i} is computational time of the task. Each edge (e_{ij}) between the tasks v_i and v_j is modelled as a tuple of the form: $\{l_{e_{ij}}, t_{e_{ij}}, E_{e_{ij}}\}$. $l_{e_{ij}}$ is the data size generated from v_i and is required to execute v_j. $E_{e_{ij}}$ and $t_{e_{ij}}$ are the communication energy consumption and communication time required to send e_{ij} from the sensor node that executes the task v_i to the sensor node that executes the task v_j.

4.2 The Wireless Sensor Network Model

The WSN is composed of a number of heterogeneous sensor nodes distributed randomly in the area of interest. The sensor nodes have different specifications, such as processing speed, power consumption and transmission distances. WSN is modelled as a graph $W = (S, D)$, where $S = \{s_x : x = 1,2,..m\}$ is the set of heterogeneous sensor nodes and $D = \{d_k : k = 1,2,..p\}$ is a set of communication links among sensor nodes. The edge $d_k \in D$ between the sensor nodes s_x and s_y is denoted as d_{xy} and is the physical distance between sensor nodes s_x and s_y.

Sensor node, s_x is modelled as a tuple of several properties and states as follows: $s_x = \{ID_{s_x}, x_{s_x}, y_{s_x}, E_r(k, s_x), f_{s_x}, e_{s_x}, a_{s_x}\}$, where ID_{s_x} is the sensor node identification, x_{s_x}, y_{s_x} are the xy coordination of sensor node, $E_r(k, s_x)$ is the battery remaining energy of sensor node at time k, e_{s_x} is the average power consumption for the processor of node (s_x), a_{s_x} is the time at which the sensor node is available to execute a task and f_{s_x} is the processing speed of sensor node. Sensor nodes s_x and s_y can directly communicate if the distance between them, d_{ij} is less than or equal to the radio range, R_r. The distance between sensor nodes (s_x and s_y) is calculated using Euclidean distance according to the following equation:

$$d_{xy} = \sqrt{(x_x - x_y)^2 + (y_x - y_y)^2}. \quad (1)$$

Therefore, at time k, the sensor node, s_x has a set of $m_{s_x}(k)$ neighbours, $N_{s_x}(k)$, where: $N_{s_x}(k) = \{s_l: \forall k\ satisfies\ d_{xl} \leq R_r\}$.

4.3 Cost Functions

4.3.1 Execution Time

The CPU clock frequency is defined as the number of computational cycles that can be executed per second. Therefore, the computational time (t_{v_i}) required to execute the task (v_i) is computed using the following formula:

$$t_{v_i} = \frac{N_{v_i}}{f_{s_x}} \quad (2)$$

where f_{s_x} is the CPU clock frequency of sensor node (s_x) which executes the task (v_i). The total computational time (called serial execution time) required to computationally execute the application tasks is the sum of computational times for all tasks and is calculated as follows:

$$t_T^p = \sum_{k=1}^{n} t_{v_k} \quad (3)$$

The communication time ($t_{e_{ij}}$) required to send the e_{ij} from the sensor node that executes the task v_i to the sensor node that executes the task v_j is made up of transmission time, queue time and propagation time. The queue time is the latency caused by media access to avoid interference and collision. Therefore, the communication time ($t_{e_{ij}}$) is computed as follows:

$$t_{e_{ij}} = t_{e_{ij}}^t + t_{e_{ij}}^p + t_{e_{ij}}^q = \frac{l_{e_{ij}}}{R_b} + \frac{d}{c} + t_{e_{ij}}^q \quad (4)$$

where $t_{e_{ij}}^t$ is the transmission time which is the data size ($l_{e_{ij}}$) divided by the data rate or communication bandwidth (R_b), $t_{e_{ij}}^p$ is the propagation time which is the distance between the sensor nodes that exchange the edge (d) divided by the speed of light ($c = 3 \times 10^8 m/s$) and $t_{e_{ij}}^q$ is the queue time. The total communication time required to exchange all the interdependences of the application tasks is the sum of all communication times required to send all dependencies and is determined as follows:

$$t_T^c = \sum_{k=1}^{q} t_{e_{ij}} \quad (5)$$

The node/task pairs are modelled as $P(v, s)$, where $P(v, s)$ shows the "n" mapped tasks of application DAG (the set) with its corresponding "$n_g(k, A_d)$" assigning sensor nodes which are the set $S_g(k, A_d)$. Hence, the overall time required to execute the application tasks using node/task pair ($P(v, s)$) is the sum of the serial execution time and total communication time and is calculated as:

$$t_T[P(v, s)] = t_T^p + t_T^c \quad (6)$$

Each task (v_i) mapped to sensor node (s_x) is starting to be executed at a time called starting executing time of the task ($t_s(v_i, s_x)$). The task is executed when the sensor node is available after it receives all the task dependencies. It is assumed that $t_{max}[pred(v_k)]$ is the time at which the last dependency (i.e., predecessor) of task (v_k) is received by the node (s_x). Therefore, after receiving the last dependency, the task (v_k) can be executed if the CPU of sensor node (s_x) is available. The time to which the sensor node (s_x) is available is referred as (a_{s_x}). Thus, $t_s(v_i, s_x)$ is the maximum of one of the two: ($t_{max}[pred(v_k)]$) or (a_{s_x}). $t_s(v_i, s_x)$ and is calculated as:

$$t_s(v_i, s_x) = max\{a_{s_x},\ t_{max}[pred(v_k)]\} \quad (7)$$

When the sensor node (s_x) starts to execute the tasks, it finishes after a time equal to the task execution time. The time at which the task is completely executed is called the finishing execution time of the task ($t_f(v_i, s_x)$), which is the time at which the task is started to be executed ($t_s(v_i, s_x)$)

plus the task execution time (t_{v_i}) and is given by:

$$t_f(v_i, s_x) = t_s(v_i, s_x) + t_{v_i} \quad (8)$$

The make-span of the application DAG is the time at which the application execution completely finishes. Due to parallelism, the make-span will be less than ($t_T[P(v,s)]$). The execution of application is completed after finishing of execution of last task. Thus, the finishing execution time of the last task will be the biggest finishing execution time. Hence, the biggest finishing execution time is the make-span and is calculated as follows:

$$ms[A_d, P(v,s)] = max_{\forall s_x \in S_g(k,A_d)}\{\, t_f(v_i, s_x)\,\} \quad (9)$$

4.3.2 Energy Consumption

The computational energy consumption (E_{v_i}) required to execute the task (v_i) is computed using the following formula:

$$E_{v_i} = e_{s_x}.t_{v_i} \quad (10)$$

where e_{s_x} is the average power consumption for the processor of node (s_x). The energy consumption ($E_{e_{ij}}$) required to send the e_{ij} from the sensor node that executes the task v_i to the sensor node that executes the task v_j is calculated as:

$$E_{e_{ij}} = E_{e_{ij}}^{TX} + E_{e_{ij}}^{RX} \quad (11)$$

where $E_{e_{ij}}^{TX}$ is the transmitted energy consumption dissipated from the source node and $E_{e_{ij}}^{RX}$ is the received energy consumption dissipated from the destination node. $E_{e_{ij}}$ is equal to zero if the tasks v_i and v_j are mapped to the same sensor node. $E_{e_{ij}}^{TX}$ and $E_{e_{ij}}^{RX}$ are calculated as follows [40]-[41]:

$$E_{e_{ij}}^{TX} = \left(e_{elec} + \varepsilon_{amp}.d^2\right).l_{e_{ij}} \quad (12)$$

$$E_{e_{ij}}^{RX} = e_{elec}.l_{e_{ij}} \quad (13)$$

where e_{elec} is the electronic energy required to transmit a bit that depends on coding, modulation and filtering and ε_{amp} is related to the radio energy. The total processing energy consumption (called serial energy consumption) required to computationally execute the application tasks is determined as follows:

$$E_T^p = \sum_{k=1}^{n} E_{v_k} \quad (14)$$

The total communication energy consumption required to exchange the interdependences of the application tasks is calculated as follows:

$$E_T^c = \sum_{k=1}^{q} E_{e_{ij}} \quad (15)$$

The overall energy consumption required to execute the application tasks using node/task pair, $P(v,s)$ is calculated as:

$$E_T[P(v,s)] = E_T^p + E_T^c \quad (16)$$

4.4 Problem Definition

At time step k, a target sensor node (s_{TSN}) triggers a request to collaboratively execute an application DAG (A_d). The number of neighbours of s_{TSN} at time step k is $n_s(k, A_d)$ and is contained in a set $S_s(k, A_d)$. $S_s(k, A_d)$ participates to execute the application. However, only $n_g(k, A_d)$ sensor nodes are selected from $S_s(k, A_d)$ to execute the application DAG (A_d). The set $S_g(k, A_d)$ includes the selected $n_g(k, A_d)$ sensor nodes. The objective function is defined as the weighted sum of the total energy consumption and the make-span. It is calculated as follows:

$$F_{obj}[A_d, P(v,s)] = \alpha * \frac{ms[A_d, P(v,s)]}{t_T^p[A_d, P(v,s)]} + (1-\alpha) * \frac{E_T[A_d, P(v,s)]}{E_{T(max)}[A_d, P(v,s)]} \quad (17)$$

where $0 \leq \alpha \leq 1$ is a weighted controlled parameter, $ms[A_d, P(v,s)]$ is the make-span to execute the

application DAG (A_d) using the mapped task/sensor ($P(v,s)$), $t_T^p[A_d, P(v,s)]$ is the serial execution time of application DAG (A_d) using the mapped task/sensor ($P(v,s)$), $E_T[A_d, P(v,s)]$ is the total energy consumption to execute the application DAG (A_d) using the mapped task/sensor ($P(v,s)$) and $E_{T(max)}[A_d, P(v,s)]$ is the maximum energy consumption to execute the application DAG A_d using the mapped task/sensor ($P(v,s)$). The make-span in Equation (17) is normalized by dividing it by the serial execution time ($t_T^p[A_d, P(v,s)]$) which is the maximum time required to execute the application. Similarly, the total energy consumption in Equation (17) is normalized by dividing it by the maximum total energy consumption ($E_{T(max)}[A_d, P(v,s)]$). The main goal is to get the task/node pair ($P^*(v,s)$) which is used to execute the application. $P^*(v,s)$ is obtained so that the objective function defined in Equation (17) is minimized according to the following objective function:

$$P^*(v,s) = arg\,min_{P(v,s)}\{F_{obj}[A_d, P(v,s)]\} \tag{18}$$

5. THE MODIFIED RANDOM BIT CLIMBING

5.1 Lifetime Awareness Sensor Node Selection Algorithm (LA-SNSA)

Awareness Sensor Node Selection Algorithm (LA-SNSA) aims to select a number of $n_g(k, A_d)$ sensor nodes from the $S_s(k, A_d)$ set. The selected nodes are then kept in the $S_g(k, A_d)$ set. In addition, the selected sensor nodes ($S_g(k, A_d)$) are used to execute the application DAG (A_d). The LA-SNSA takes into account the network lifetime. Since reducing the gaps which appear because of node death in the network increases the network lifetime, the sensor nodes with higher number of neighbours are preferred to be selected. Furthermore, LA-SNSA also takes into account the current remaining energy of the sensor nodes. Thus, sensor nodes with higher remaining energy are favoured to be selected to increase the network lifetime. Therefore, the objective function of the LA-SNSA is the weighted sum of the ratio of energy of sensor node with respect to the sum of remaining energy for all nodes in $S_s(k, A_d)$ and the ratio of the number of neighbours of the sensor node with respect to the sum of the number of neighbours for all nodes in $S_s(k, A_d)$. It is computed as follows:

$$F_{obj}(k, s_x, A_d) = \beta * \frac{E_r(k,s_x)}{\sum_{\forall s_l \in S_s(k,A_d)} E_r(k,s_l)} + (1-\beta) * \frac{N_c(k,s_x)}{\sum_{\forall s_l \in S_s(k,A_d)} N_c(k,s_l)} \tag{19}$$

As seen in the above formula, β is a weighting parameter and varies in the interval [0, 1]; and $N_c(k, s_x)$ is the number of neighbours of sensor node s_x at time k . Algorithm 1 shows the LA-SNSA. The weighting parameter (β) is firstly selected. Then, the objective function for sensor nodes in $S_g(k, A_d)$ set is calculated based on Equation (19). After that, a number of $n_g(k, A_d)$ sensor nodes, with the highest objective function, are selected and added to $S_g(k, A_d)$ set.

Algorithm 1: Lifetime Awareness Sensor Node Selection Algorithm (LA-SNSA)

select β;
set $\ell = 0$;
while $\ell \leq n_g(k, A_d)$ **do**:
  for each sensor node $s_x \in \mathbf{S}_s(k, A_d)$ **do**:
    calculate $F_{obj}(k, s_x, A_d)$ based on Equation (19);
  end for;
  find the sensor s_x^* with maximum $F_{obj}(k, s_x, A_d)$;
  add s_x^* to $\mathbf{S}_g(k, A_d)$;
  remove s_x^* from next search;
  increment ℓ: $\ell = \ell + 1$;
end while;

5.2 Random Bit Climbing (RBC)

Random Bit Climbing (RBC) optimization [42]-[43] is a metaheuristic local search-based algorithm that employs a trajectory-based approach to guide the search and obtain a (near) optimal solution.

RBC is a single-solution metaheuristic, which adopts the exploitation in its operation through memorizing the best current solution. In RBC, single stochastic solution is used for each round. Firstly, an initial single-parent (p) is randomly generated and set as the current solution. After that, the objective function of the initial parent is evaluated. Then, a random arrangement of the index positions for the current solution is created and kept in the π vector. Next, a child is produced by mutating a single dimension of the current solution at a time. The child replaces the current solution if it fulfils the objective function. The evaluation of children either continues for all possible children or is terminated when the first better child is found. Then, a new random permutation is generated for the current solution. The process continues until a predefined number of iterations have been exhausted. However, the main limitation of RBC is the trap of local optimal solution because of its deficiency for exploration ability.

5.3 λ-Modified Random Bit Climbing (λ-mRBC)

Algorithm 2 shows the proposed λ-Modified Random Bit Climbing (λ-mRBC). Because the exploration is tied up to randomness [17], the λ-mRBC adopts a random parameter (λ) to use exploration in RBC operation. The solution is represented as a vector of n elements. The vector index represents the task number (from 1 to n). On the other hand, the vector value represents one of the selected sensor node numbers. In Step (1), an initial parent solution ($P(v,s)$) is generated randomly. This initial parent solution is then set as the current best solution and is stored in $C_s(v,s)$. The evaluation is performed in Step (2) to calculate the fitness value of the current best solution. In Step (3), the random permutation for the current best solution is achieved to produce the permutation vector (π). In Step (4), a new operator named random transposition operator (trans) is added into the RBC method to escape from local optima and to increase the exploration of the search space. The random transposition operator is performed on the current solution according to the following rule:

$$C_s(v,s) = \begin{cases} trans\left(C_s(v,s)\right) & r < \lambda \\ C_s(v,s) & otherwise \end{cases} \quad (20)$$

where r is a random number which uniformly distributes between [0, 1], λ an algorithm parameter number which ranges between 0 and 1 and trans is the transposition operation that randomly exchanges the places of the current best solution. The children are generated in Step (5) by cloning $C_s(v,s)$ and flipping the position π_l. After that, the child is evaluated in Step (6). In Step (7), the child replaces the current best solution if it has a better fitness value. In Step (8), the children are generated and evaluated. The algorithm flow continues to the next iteration in Step (9). The operations are repeated until the maximum iterations are exhausted in Step (10). After termination, the current best solution is returned as the suboptimal solution of the problem.

Algorithm 2: λ-Modified Random Bit Climbing (λ-mRBC)

Step (1) Compute the initial parent task/node pairs $P(v,s)$; set the current best solution $C_s(v,s) = P(v,s)$; and set $iter = 1$.

Step (2) Calculate the fitness value $F_{obj}[A_d, C_s(v,s)]$ of $C_s(v,s)$.

Step (3) Generate the random permutation $\boldsymbol{\pi} = (\pi_1, \pi_2 \ldots \pi_m)$ of the position of $C_s(v,s)$; and set $l = 1$.

Step (4) if (r < λ): execute transposition operation of $C_s(v,s)$ positions.

Step (5) Generate the child (offspring) $O_l(v,s)$ by cloning $C_s(v,s)$ and flipping the position π_l;

Step (6) Calculate the objective function $F_{obj}[A_j, O_l(v,s)]$ of $O_l(v,s)$.

Step (7) If ($\boldsymbol{F_{obj}[A_d, O_l(v,s)] < F_{obj}[A_d, C_s(v,s)]}$ **):** replace $C_s(v,s) = O_l(v,s)$; and go to Step (9).

Step (8) If ($\boldsymbol{l > m}$**):** Go to Step (9)

else: Increment l: $l = l + 1$; and go to step (5);

Step (9) Increment $iter = iter + 1$.

Step (10) If ($\boldsymbol{iter > max\,iter}$ **):** go to step (11)

else: go to Step (3)

Step (11) Return C_s as the suboptimal solution and finish.

5.4 The Complete TMS Approach

Figure 4 explains the proposed TMS approach. First of all, heterogeneous sensor nodes are created and WSN is randomly distributed. Then, the algorithm parameters for λ-mRBC and LA-SNSA are set and defined. When a target sensor node requests execution of an application, a DAG of the requested application is created. LA-SNSA is performed based on Algorithm 1 to select the sensor nodes that will cooperatively execute the application so that the network lifetime is improved. λ-mRBC is achieved based on Algorithm 2 to optimally get the best task/node pairs with minimum execution time and energy consumption. After that, λ-mRBC method is repeated until termination condition is met. Finally, the final solution of task/node pair is obtained and simulation statistics are recorded.

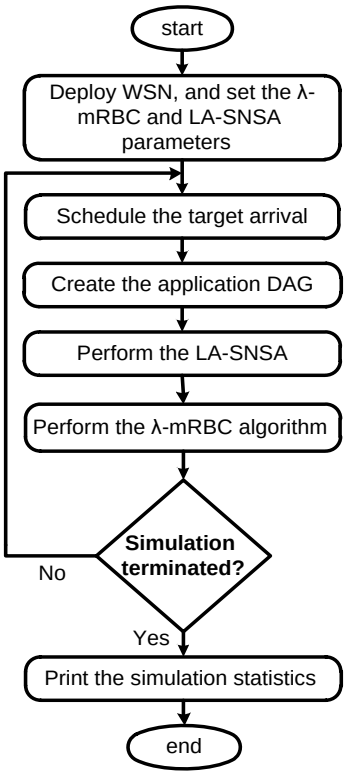


Figure 4. The proposed TMS approach.

6. SIMULATION RESULTS

This section evaluates the proposed λ-mRBC method. C++ is used to build the simulation environment using core i5 of 2.5 GHz processor and 4 GB memory.

6.1 Simulation Setting

6.1.1 The Parameters for Application DAG

Unless it is clearly stated, the application DAG consists of fifteen tasks ($n = 15$) as follows: four tasks are used as entry tasks, ten tasks are used as normal tasks and one task is used as an exit task. The immediate successors for each entry and normal tasks are selected to be uniformly distributed in the range of [1, 3]. The computation load of each task (N_{v_i}) is initialized to be uniformly distributed in the range of [300, 600] Kilo Clock Cycles (KCC). The communication load for edges among tasks (e_{ij}) is initialized to be uniformly distributed in the range of [500, 800] bytes of data.

6.1.2 The Parameters for WSN

WSN is implemented using 224 heterogeneous sensor nodes which are deployed randomly in a monitoring region of 200 m × 200 m. The transmission radio is set to $R_r = 100$ m. The radio channel with bandwidth (i.e., bit rate) of 250 Kbps is used in the simulation environment. The processing speed for sensor nodes (f_{s_x}) stands for the total number of clock cycles which can be executed within

one second. It is set to be uniformly distributed in the range of [30, 100] Million Cycles per Second (MCPS).The power consumption of the processors for sensor nodes (e_{s_x}) is set to be uniformly distributed in the range of [4, 10] mJ. The initialized energy level ($E_r(0, s_i)$) of each sensor is set to be uniformly distributed in the range of [0, 1] J.

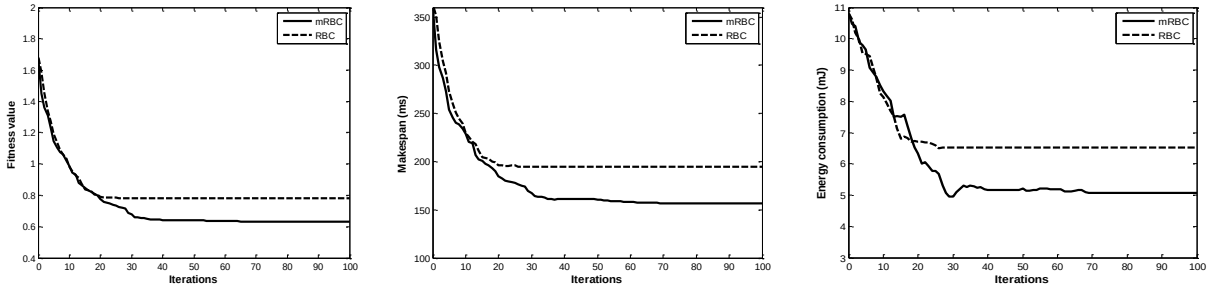
6.1.3 The Parameters for the λ-mRBC

The weighting parameters are set as follows: β=0.5 and α=0.5. The number of iterations for λ-mRBC is assumed to be 100. Unless it is clearly stated, the number of selected nodes to execute the application is $n_g(k, A_d) = 3$ sensor nodes. The λ-mRBC algorithm parameter λ is set to 0.5.

6.2 Results and Analysis

6.2.1 Impact of Number of Iterations

The fitness value of the best solution is plotted in Figure 5 (a) for the RBC and λ-mRBC methods. It becomes clear that the RBC method has a lower convergence speed, compared with the proposed λ-mRBC method. Additionally, the RBC is trapped in local minima. On the other hand, the proposed λ-mRBC method converges to better fitness value by 19.1%, compared with RBC method. This is because of using the transposition operator (trans), where the positions of current best solution elements are randomly swapped. The transposition operator (trans) which is controlled by adjusting the λ parameter occurs in some selected iterations. When the elements of the current best solution are randomly swapped, more exploration in the search space occurs. Hence, the λ-mRBC method tries to escape from the trap of local minima. Consequently, better solution can be found. Figure 5 (b) and Figure 5 (c) show the make-span and total energy consumption *versus* iteration for both RBC and λ-mRBC methods. Compared with RBC method, the proposed λ-mRBC method converges to better make-span and total energy consumption by 19.6% and 22.3%, respectively. Since the fitness value of λ-mRBC method has better convergence speed and lower values, the performance of the proposed λ-mRBC method, in terms of make-span and total energy consumption, is improved, compared with the RBC method.



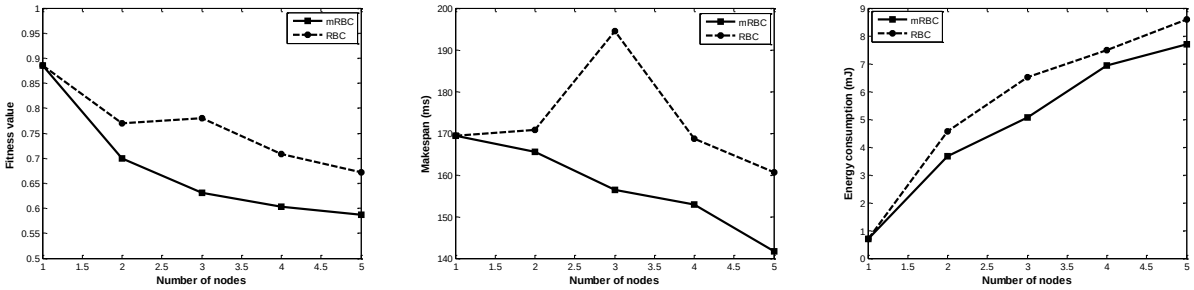
(a) Fitness value (b) Make-span (c) Energy consumption

Figure 5. The effect of iterations for the RBC and proposed λ-mRBC methods.

6.2.2 Impact of Varying Number of Sensor Nodes

This section evaluates the effects of selected sensor node size ($n_g(k, A_d)$). The proposed λ-mRBC method supports different sizes of the sensor nodes. The size of the sensor nodes is changed from 1 to 5 with one sensor node for each step. Figure 6 shows the performance of the RBC and proposed λ-mRBC methods with the sensor node size. As shown in Figure 6 (a), the fitness value is getting better whenever the size of the selected sensor nodes increases. This is because the computational load of tasks is parallelized in more powerful fashion whenever the size of the selected sensor nodes rises. However, the proposed λ-mRBC method gives lower fitness values, compared with traditional RBC method. In addition, the fitness value of RBC method at a sensor node size of 3 does not improve, compared with its value at a sensor node size of 3. This is due to the trapping in the local minima. The make-span shown in Figure 6 (b) is reduced whenever the sensor node size goes up, because the computational load is distributed to more sensor nodes. As shown in Figure 6 (c), the communication activities used to exchange the communication edges increase whenever the sensor node size rises up, because tasks can be distributed to more sensor nodes. Therefore, according to Equation (15), the total

energy consumption rises with increasing the sensor node size. Ultimately, the proposed λ-mRBC method improves the energy consumption. Additionally, compared with RBC method, the proposed λ-mRBC method has better fitness value, make-span and total energy consumption by 11.8% , 10.3% and 12.6%, respectively.

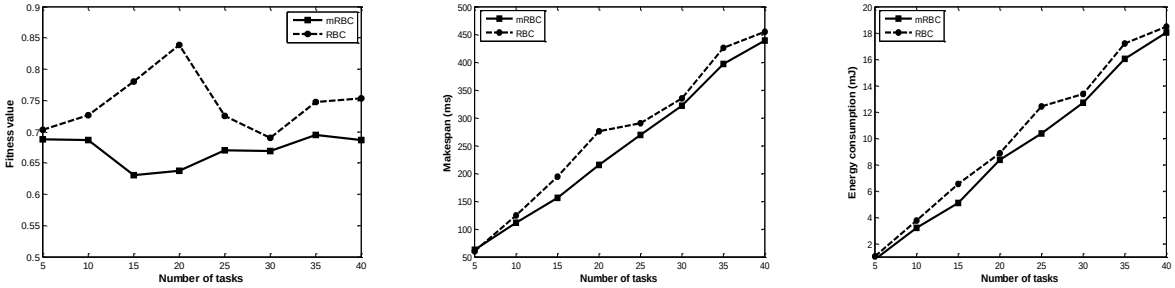


(a) Fitness value (b) Make-span (c) Energy consumption

Figure 6. The effect of node size for the RBC and proposed λ-mRBC methods.

6.2.3 Impact of Number of Tasks

The proposed λ-mRBC method supports different numbers of tasks. In Figure 7, the number of tasks varies from 5 to 40 with five tasks for each step. The fitness value, make-span and energy consumption for each step are plotted. In fact, according to Equation (14) and Equation (15), increasing the number of tasks leads to the increasing of the computational and communicational loads. Therefore, make-span and energy consumption increase with increasing the number of tasks. This is shown in Figure 7 (b) and Figure 7 (c). The aim of the objective function of Equation (17) is to reduce the energy consumption and the make-span as well. Thus, some solutions give better improvement in terms of energy consumption and other solutions give improvement in terms of make-span. Therefore, the fitness values shown in Figure 7 (a) fluctuate with increasing the number of tasks. It is worth mentioning that the proposed λ-mRBC method can cope with different numbers of tasks due to the small fluctuation of fitness values, compared with the RBC method. Besides, λ-mRBC method gives better performance in terms of make-span and energy consumption. Furthermore, compared with RBC method, the proposed λ-mRBC method has better fitness value, make-span and total energy consumption by 3.6%, 2.4% and 8.8%, respectively.



(a) Fitness value (b) Make-span (c) Energy consumption

Figure 7. The effect of varying number of tasks for the RBC and proposed λ-mRBC methods.

6.2.4 LA-SNSA Evaluation

This section evaluates the performance of LA-SNSA. The performance metrics used to evaluate the proposed LA-SNSA are the Remaining Energy Performance (*REP*) and the Neighbour Count Performance (*NCP*). *REP* is defined as the normalized sum of normalized remaining energy of all sensor nodes and *NCP* is defined as the normalized sum of the number of neighbours of all sensor nodes. Therefore, *REP* and *NCP* are calculated as follows:

$$REP(k) = \frac{\sum_{l=1}^{m}[E_r(k,s_l)/E_{r(max)}(k,s_l)]}{REP_{max}} \quad (21)$$

$$NCP(k) = \frac{\sum_{l=1}^{m}[N_c(k,s_l)]}{NCP_{max}} \quad (22)$$

Where REP_{max} and NCP_{max} are the maximum values of *REP* and *NCP* which are calculated at the beginning of the simulation. Therefore, $REP(k)$ and $NCP(k)$ values are in the range of [0, 1]. In Figure 8 and Figure 9, the proposed LA-SNSA and random selection schemes are compared by calculating these performance metrics. In random selection scheme, however, the sensor nodes ($n_g(k, A_d)$) are chosen randomly from the neighbouring target sensor node (s_{TSN}).

As shown in Figure 8, the $REP(k)$ is decreasing with time. This is due to the increasing energy consumption of communication and processing activities, which are caused by the application executions. It is observed from Figure 8 that the rate of $REP(k)$ reduction with time using the proposed LA-SNSA is smaller than those in the random selection scheme. The reason behind this is that LA-SNSA aims to select the nodes with higher remaining energy, while the random selection scheme selects the sensor nodes randomly without any knowledge of the node remaining energy. The proposed LA-SNSA enhances the $REP(k)$ about three times, compared with the random selection scheme.

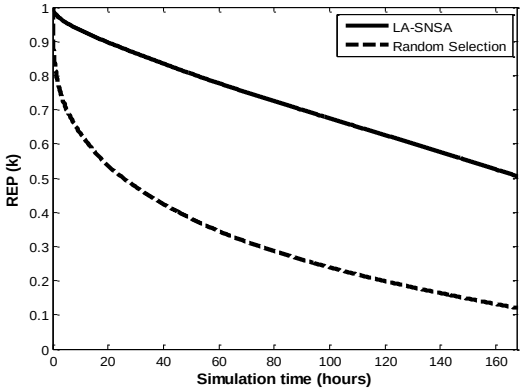


Figure 8. REP for the RBC and proposed λ-mRBC methods.

The value of $NCP(k)$ remains 1 until the first node death takes place. Thus, $REP(k)$ and number of dead nodes are calculated and plotted in Figure 9 after the death of the first node. Application executions lead to energy consumption, caused by processing and communicating. Therefore, sensor node energy level decreases. When the energy level of sensor node is exhausted, the sensor node dies and all activities stop. After the first death, $NCP(k)$ is decreased due to the increasing of death nodes. As shown in Figure 9, $NCP(k)$ is decreased sharply in case of random selection scheme, because there are no directional guides to select sensor nodes. Additionally, the rate of increased dead nodes is higher in case of random selection scheme. Another advantage of the proposed LA-SNSA is that it takes a long time for first node to die. Furthermore, compared with random selection, the $NCP(k)$ is improved by 20.1% using the proposed LA-SNSA.

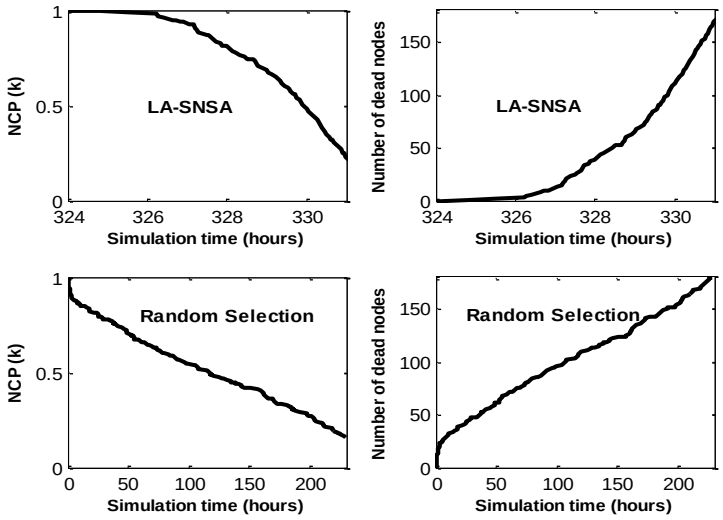


Figure 9. NCP and number of dead nodes for the RBC and proposed λ-mRBC methods.

Table 1 shows the Remaining Energy Performance (*REP*), Neighbour Count Performance (*NCP*) and first node death time for the proposed LA-SNSA and random selection schemes. The proposed LA-SNSA ends with better *REP* which implies that the remaining energy of the sensor nodes is better distributed in the network. In addition, there is an improvement in *NCP* in case of using the proposed LA-SNSA. This improvement leads to less gaps without sensor nodes in the network. The network lifetime can be defined as the time when the first node dies [44]-[45]. First node death time is bigger in case of the proposed LA-SNSA. Since *REP*, *NCP and* first node death time are improved in the proposed LA-SNSA scheme, the network lifetime is also enhanced in case of using the proposed LA-SNSA.

Table 1. REP, NCP and first node death time for random and LA-SNSA schemes.

Method	Remaining Energy Performance (*REP*)	Neighbour Count Performance (*NCP*)	First Node Death time (Hours)
LA-SNSA	0.51	0.19	324.79
Random	0.12	0.16	0.03

6.2.5 The Effect of λ-mRBC Parameter

Figure 10 shows the fitness values *versus* the iteration using λ parameter values of 0.1, 0.3, 0.5, 0.7 and 0.9. According to Equation (20), the probability of running the transposition operation is increasing with increasing the λ parameter. Therefore, the convergence speed for λ parameter of 0.1 and 0.3 is the slowest, compared with other λ parameter values. Furthermore, when using λ parameters of 0.1 and 0.3, the λ-mRBC method converges to the highest fitness value. On the other hand, when using λ parameters of 0.5 and 0.7, the λ-mRBC method converges to the lowest fitness value. The fastest convergence speed occurs when using λ parameter of 0.9. However, λ-mRBC method converges to larger fitness value than the fitness value when using λ parameters of 0.5 and 0.7. It is worth mentioning that the number of optimization algorithm parameters increases the complexity of the algorithm [17]. λ-mRBC uses only one parameter (λ) which indicates its low complexity.

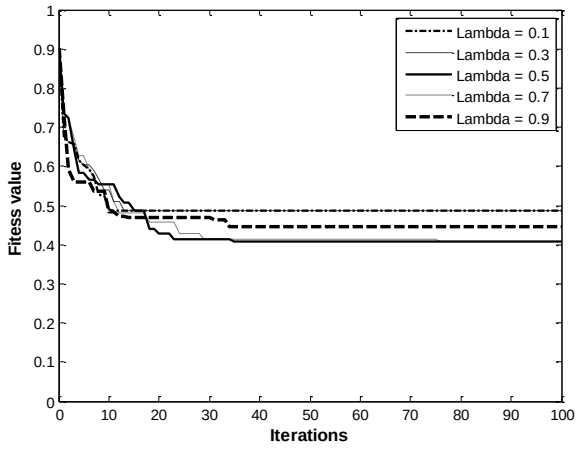


Figure 10. The effect of λ parameter.

7. CONCLUSION

In this paper, a Task Mapping and Scheduling (TMS) approach for WSN is introduced to look for the best tasks/nodes mapping solution. The proposed λ-mRBC, which is a modified version of RBC optimization method, is proposed to improve the performance of the search. To escape from local optima and to increase the exploration of the search space, the λ-mRBC method employs a new operator, which is named random transposition. The transposition operator changes the elements' positions of current best solution. The λ-mRBC method is controlled by using only one parameter (λ). Energy consumption and application execution time (make-span) are taken into consideration in the fitness objective to get the best performance of TMS. In addition, LA-SNSA is proposed to select a

number of sensor nodes needed to execute the applications, so that the network lifetime is improved. The simulation results show that the proposed λ-mRBC method improves the energy consumption, make-span and fitness value, compared with traditional RBC method. Furthermore, using LA-SNSA enhances the network lifetime, compared with random selection approaches. Although the proposed λ-mRBC uses a new operator called transposition operator to escape from local optima, it is still a single-solution metaheuristic. Unlike population-based metaheuristics, the proposed λ-mRBC is of less exploration of search space. The future work aims to add a new operator that employs more than one solution to increase the exploration of search space.

REFERENCES

[1] B. Sharma and T. C. Aseri, "A Comparative Analysis of Reliable and Congestion-aware Transport Layer Protocols for Wireless Sensor Networks," International Scholarly Research Network (ISRN), Sensor Networks, 2012.

[2] M. Katiyar, H. P. Sinha and D. Gupta, "On Reliability Modeling in Wireless Sensor Networks-A Review," IJCSI International Journal of Computer Science, vol. 9, no. 6, pp. 134–146, 2012.

[3] H. Yetgin, K. T. K. Cheung, M. El-Hajjar and L.H. Hanzo, "A Survey of Network Lifetime Maximization Techniques in Wireless Sensor Networks," IEEE Communication Surveys & Tutorials, vol. 19, no. 2, pp. 828-854, 2017.

[4] P. R. Pereira, A. Grilo, F. Rocha, M. S. Nunes, A. Casaca, C. Chaudet, P. Almstrom and M. Johansson, "End to End Reliability in Wireless Sensor Networks: Survey and Research Challenges," Proceedings of the EuroFGI Workshop on IP QoS and Traffic Control, 2007.

[5] M. A. Kafi, J. B. Othman, M. Bagaa and N. Badache, "CCS_WHMS: A Congestion Control Scheme for Wearable Health Management System," Journal of Medical Systems, vol. 39, no. 12, 2015.

[6] Y. E. Hamouda and M. M. Msallam, "Smart Heterogeneous Precision Agriculture Using Wireless Sensor Network Based on Extended Kalman Filter," Neural Computing and Applications, pp.1-17, 2018.

[7] P. R. C. Araújo, R. H. Filho, J. J. Rodrigues, J. P. Oliveira and S. A. Braga, "Middleware for Integration of Legacy Electrical Equipment into Smart Grid Infrastructure Using Wireless Sensor Networks," Inter.

Journal of Communication Systems, vol. 31, no. 1, pp. e3380, 2018.

[8] B. L. R. Stojkoska and K. V. Trivodaliev, "A Review of Internet of Things for Smart Home: Challenges and Solutions," Journal of Cleaner Production, vol. 140, no. 3, pp.1454-1464, 2017.

[9] Y. E. Hamouda and C. Phillips, "Adaptive Sampling for Energy-efficient Collaborative Multi-Target Tracking in Wireless Sensor Networks," IET Wireless Sensor Systems, vol. 1, no. 1, pp.15-25, 2011.

[10] J. Luo and S. Zou, "Strong k-barrier Coverage for One-way Intruders Detection in Wireless Sensor Networks," International Journal of Distributed Sensor Networks, vol. 12, no. 6, 2016.

[11] J. Manikannu and V. Nagarajan, "A Survey of Energy Efficient Routing and Optimization Techniques in Wireless Sensor Networks," IEEE International Conference on Communication and Signal Processing (ICCSP), 2018.

[12] M. Wolf, Smart Camera Design, Springer, 2018.

[13] M. Karakaya and H. Qi, "Coverage Estimation in Heterogeneous Visual Sensor Networks," Proceedings of the 8th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 41-49, 2012.

[14] C. A. Navarro, N. Hitschfeld-Kahler and L. Mateu, "A Survey on Parallel Computing and Its Applications in Data-parallel Problems Using GPU Architectures," Communications in Computational Physics, vol. 15, no. 2, pp. 285-329, 2014.

[15] L. Dai, H. Xu, T. Chen, Q. Chao and L. Xie, "A Multi-Objective Optimization Algorithm of Task Scheduling in WSN," International Journal of Computers, Communications & Control, vol. 9, no. 2, pp. 160-171, 2014.

[16] Y. Yang, X. Qiu, L. Meng and K. Long, "Task Coalition Formation and Self-adjustment in the Wireless Sensor Networks,". Int J. Commun. Syst., vol. 27, no. 10, pp. 2241–2254, 2014.

[17] C. Blum and A. Roli,"Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," ACM Computing Surveys (CSUR), vol. 35, no. 3, pp. 268-308, 2003.

[18] I. Boussaïd, J. Lepagnot and P. Siarry, "A Survey on Optimization Metaheuristics," Information Sciences, vol. 237, pp. 82-117, 2013.

[19] Y. Jin, J. Jin, A. Gluhak, K. Moessner and M. Palaniswami, "An Intelligent Task Allocation Scheme for Multihop Wireless Networks," IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 3, pp. 444-451, 2012.

[20] R. Shams and F. Khan, "Solving Wireless Network Scheduling Problem by Genetic Algorithm," IAMURE International Journal of Mathematics Engineering & Technology, vol. 2, no. 11, pp. 63-70, 2012.

[21] J. Yang, H. Zhang, Y. Ling, C. Pan and W. Sun, "Task Allocation for Wireless Sensor Network Using Modified Binary Particle Swarm Optimization," IEEE Sensors Journal, vol. 14, no. 13, pp. 882-892, 2014.

[22] A. A. Ferjani, N. Liouane and I. Kacem, "Task Allocation for Wireless Sensor Network Using Logic Gate-based Evolutionary Algorithm," International Conference on Control, Decision and Information Technologies (CoDIT), pp. 654-658, 2016.

[23] V. Papataxiarhis, "Optimal Task Assignment in Sensor Networks," Proc. of the 17th IEEE International Conference on Mobile Data Management (MDM), pp. 26-31, 2016.

[24] S. Abdelhak, C. S. Gurram, S. Ghosh and M. Bayoumi, "Energy-balancing Task allocation on Wireless Sensor Networks for Extending the Lifetime," Proceedings in the 53rd IEEE Int. MWSCAS, pp. 781-784, 2010.

[25] X. Yin, W. Dai, B. Li, L. Chang and C. Li, "Cooperative Task Allocation in Heterogeneous Wireless Sensor Networks", Inter. Journal of Distributed Sensor Networks, vol. 13, no. 10, pp. 1-12, 2017.

[26] D. R. Bolla, J. J. Jijesh and M. S. Pramod, "Real-Time Data Fusion Applications in Embedded Sensor Network Using TATAS," Indian Journal of Science and Technology, vol. 10, no. 13, pp. 1-7, 2017.

[27] Y. Tian and E. Ekici, "Cross-Layer Collaborative in Network Processing in Multihop Wireless Sensor Networks," IEEE Trans. Mobile Comput., vol. 6, no. 3, pp. 297-310, 2007.

[28] Y. Tian, B. Jarupan, E. Ekici and F. Ozguner, "Real-Time Task Mapping and Scheduling for Collaborative in Network Processing in DVS-Enabled Wireless Sensor Networks," Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006), pp. 1-10, 2006.

[29] Y. E. M. Hamouda and C. Phillips, "Biological Task Mapping and Scheduling in Wireless Sensor Network," Proc. of the IEEE International Conference on Communications Technology and Applications, pp. 914 – 919, 2009.

[30] Y. E. M. Hamouda, "Light Allocation of Tasks in Clustered-based Wireless Sensor Networks", Al-Aqsa University Journal (Natural Sciences Series), vol. 21, pp. 90-119, 2017.

[31] K. N. Devi and R. Muthuselvi, "Parallel Processing of IoT Health Care Applications," Proc. of the 10th IEEE International Conference on Intelligent Systems and Control (ISCO), pp. 1-6, 2016

[32] J. Jiang, G. Han and C. Zhu, "A Complicated Task Solution Scheme Based on Node Cooperation for Wireless Sensor Networks," Proc. of the 22nd IEEE International Conference on Parallel and Distributed Systems (ICPADS), pp. 264-269, 2016.

[33] P. Skocir, M. Kusek and G. Jezic, "Energy-efficient Task Allocation for Service Provisioning in Machine-to-Machine Systems," Concurrency and Computation: Practice and Experience, vol. 29, no. 23, pp. e4269, 2017.

[34] X. Yin, K. Zhang, B. Li, A. K. Sangaiah and J. Wang, "A Task Allocation Strategy for Complex Applications in Heterogeneous Cluster–based Wireless Sensor Networks," International Journal of Distributed Sensor Networks, vol. 14, no. 8, 2018.

[35] E. A. Khalil, S. Ozdemir and S. Tosun, "Evolutionary Task Allocation in Internet of Things-based Application Domains," Future Generation Computer Systems, vol. 86, pp.121-133, 2018.

[36] W. Yu, Y. Huang, E. Ding and A. Garcia-Ortiz, "Joint Task Allocation Approaches for Hierarchical Wireless Sensor Networks," Proc. of the IEEE 7th International Conference on Modern Circuits and Systems Technologies (MOCAST), pp. 1-4, 2018.

[37] N. Bulusu, J. Heidemann and D. Estrin, "GPS-less Low-Cost Outdoor Localization for Very Small Devices," IEEE Personal Communications, vol. 7, no. 5, pp. 28-34, 2000.

[38] T. C. Karalar, S. Yamashita, M. Sheets and J. Rabaey, "A Low-Power Localization Architecture and

System for Wireless Sensor Networks," IEEE Workshop on Signal Processing Systems, USA: Signal Processing Society, pp. 89-94, 2004.

[39] O. Sinnen, Task Scheduling for Parallel Systems, New Jersey: John Wiley & Sons, Inc., Hoboken, 2007.

[40] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient Communication Protocol for Wireless Microsensor Networks," Proceedings of the IEEE 33rd Annual Hawaii International Conference on System Sciences (HICSS '00), pp. 1-10, 2000.

[41] A. Wang and A. Chandrakasan, "Energy-efficient DSPs for Wireless Sensor Networks," IEEE Trans. Signal Process. Mag., pp. 68-78, 2002.

[42] L. Davis, "Bit-Climbing, Representational Bias and Test Suite Design," Proc. of the 4th International Conference on Genetic Algorithms, pp. 18-23, 1991.

[43] H. Aguirre and K. Tanaka, "Random Bit Climbers on Multiobjective MNK-Landscapes: Effects of 49 and Population Climbing," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. 88, pp. 334-345, 2005.

[44] M. A. Abd, S. F. M. Al-Rubeaai, B. K. Singh, K. E. Tepe and R. Benlamri, "Extending Wireless Sensor Network Lifetime with Global Energy Balance," IEEE Sensors Journal, vol. 15, no. 9, pp. 5053–5063, 2015.

[45] I. Dietrich and F. Dressler, "On the Lifetime of Wireless Sensor Networks," ACM Trans. Sen. Netw., vol. 5, no. 1, 2009.

ملخص البحث:

تعالج هذه الورقة مشكلة تخطيط المهام وجدولتها في شبكات المجسّات اللاسلكية. هذا التطبيق المفترض أن ينفَّذ في شبكات المجسّات اللاسلكية يمكن تقسيمه الى مهام يعتمد بعضها على بعض. وتتمثل الأهداف الأساسية لتخطيط المهام وجدولتها في تحسين زمن التنفيذ واستهلاك الطاقة وعُمر الشبكة. تم تطوير طريقة معدَّلة تقوم على التسلق العشوائي من أجل الحصول على حلول أفضل وأسرع تكون مثالية أو قريبة من المثالية.

في الطريقة المعدّلة المقترحة، يُضاف عامل جديد يسمى عامل الإزاحة من أجل تحسين استكشاف فضاء البحث ومن ثم الهروب من القيم المثلى المحلية. ويتم التحكم بعمق الاستكشاف باستخدام متغير واحد (λ). في البدء، يتم اختيار عدد من عُقد المجسّات من أجل التنفيذ التعاوني للتطبيق بهدف تحسين عُمر الشبكة. بعد ذلك، يتم تنفيذ الطريقة المعدّلة المقترحة للحصول على الحلّ الأمثل أو القريب من الأمثل فيما يتعلق بأزواج المهامّ / المجسّات، بحيث يتم التقليل الى الحد الأدنى الممكن من زمن التنفيذ واستهلاك الطاقة.

وقد بينت نتائج المحاكاة أن الطريقة المعدّلة المقترحة تُحسّن أداء تخطيط المهام وجدولتها. ومقارنة بالطريقة التقليدية، فإن الطريقة المقترحة تؤدي الى تحسُّن قيمة الملاءمة وفترة الفِعل والاستهلاك الكلي للطاقة بنسبٍ بلغت 19.1% و 19.6% و22.3% على الترتيب. من جانب آخر، تمت إطالة عُمر الشبكة عبر استخدام خوارزمية الاختيار المقترحة. وقد تحسّن توزيع الطاقة المتبقية بين عُقد المجسّات بمقدار 3 مرات تقريباً مقارنة بطريقة الاختيار العشوائي. علاوة على ذلك، ومقارنة بالاختيار العشوائي، فقد تحسّن عدد الجيران لعُقد المجسّات بنسبة 20.1% باستخدام خوارزمية الاختيار المقترحة.

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 05, No. 01, April 2019.

AN IMPROVED C4.5 MODEL CLASSIFICATION ALGORITHM BASED ON TAYLOR'S SERIES

I. I. Sinam[1] and Abdulwahab Lawan[2]

(Received: 3-Jan.-2019, Revised: 25-Feb.-2019, Accepted: 11-Mar.-2019)

ABSTRACT

C4.5 is one of the most popular algorithms for rule base classification. Many empirical features in the algorithm exist, such as continuous number categorization, missing value handling and over-fitting. However, despite its promising advantage over the Iterative Dichotomiser 3 (ID3), C4.5 has the major setback of presenting the equivalent result as the ID3, especially when the same number of attributes is used. This paper proposes a technique that will handle the setback reported in C4.5. The performance of the proposed technique is measured based on better accuracy. The Entropy of Information Theory is measured to identify the central attribute for the dataset. The researchers apply exponential splitting information (EC4.5) in utilizing the central attribute of the same dataset. The result obtained on introducing Taylor series suggested a far better result than when the C4.5 (gain ratio) was introduced.

KEYWORDS

ID3 Algorithm, C4.5 Algorithm, Information gain, Entropy, Gain ratio.

1. INTRODUCTION

Decision tree, as the name implies, is a predictive model that can be viewed as a tree structure, where specifically each branch of the tree is a classification question and the leaves of the tree are partitions of the dataset with their classification [1]-[2]. It is a logical model represented as a binary or multiclass tree that shows how the value of a target variable can be predicted by using the values of a set of predictor variables. Decision tree classifiers are considered "white box" classification models, as they can provide the explanation for their classification models and can be used directly for decision -making [3]. Many decision tree algorithms exist and these include: Alternating Decision Tree (LAD), C4.5 or J48 Pruned Tree, Classification and Regression Tree (CART), Chi-squared Automatic Interaction Detection (CHAID), Quest, …etc. Decision tree algorithms such as C4.5 had been developed earlier and continue to be regularly used in solving everyday classification tasks. However, despite its promising advantage over the ID3 algorithm, C4.5 has the major setback of presenting the equivalent result as the ID3, especially when the same number of attributes is used. In this paper, the predictive performance of this algorithm is enhanced by proposing another technique that will handle the noticeable setback and even present a more promising result than the C4.5 using (gain ratio). It is on this background that the exponential modification of the gain ratio is being proposed.

2. RELATED WORK

ID3 tree algorithm was introduced in 1986 by Quinlan Ross. It is based on Hunt's algorithm and the algorithm is serially implemented. The ID3 uses an information gain measure in choosing the splitting attribute [4]. The basic strategy in ID3 is the selection of splitting attributes with the highest information gain. That is; the amount of information associated with an attribute value that is related to the probability of occurrence [2]. Once the attribute has been chosen, the amount of information is measured, which is known as entropy [5]. Entropy is used to measure the amount of uncertainty, surprise or randomness in a dataset. The entropy will be zero when all the data in the set belong to the single class. One of the challenges with this approach is when ID3 selects the attribute having more number of values, which may necessarily not be the best attribute [5]. When testing a small sample, data may be over-fitted or over-classified. At a time, only one attribute is used for the testing purpose. As specified

1. I. I. Sinam is with Department of Computer Science, Bayero University, Kano, Nigeria.
2. A. Lawan is with Department of Information Technology, Bayero University, Kano, Nigeria. Email: abd_wahhb@yahoo.com

above, continuous data is difficult to analyze, as many trees need to be generated to find the perfect place to split the data, which makes the algorithm computationally expensive. The mathematical model of C4.5 is given by Equation (1).

$$Gain(P, A) = Entropy(P) - \sum_{j=1}^{v}((p_j)Entropy(p_j)) \quad (1)$$

On the other hand, C4.5 algorithm is an extension of ID3 algorithm. It has an enhanced method of tree pruning that reduces miss-classification errors due to noise or too much detail in the training dataset found in ID3. It uses the gain ratio impurity method to evaluate the splitting attribute [2], [6]. Quinlan Ross introduced split information to information gain of ID3 as an improvement to overcome the limitations of ID3, which are latency and over-fitting and it becomes computationally expensive in handling continuous data. The gain ratio is given by Equation (2).

$$Gain_{Ratio(D,A)} = \frac{Enropy(D)\sum_{j=1}^{i}\left(p_j * Enropy(p_j)\right)}{Splitting_{Info}} \quad (2)$$

i. It will increase the performance when the number of attributes differs.
ii. It will increase the performance when the number of attributes is the same.
iii. And it will decrease the percentage of uncertainty in C4.5 algorithm.

3. METHODOLOGY

To overcome the limitations of C4.5, the researchers used Taylor's Series to modify the splitting information of C4.5.

3.1 Data Collection

The study uses an existing instructor's performance dataset from Abubakar Tafawa Balewa University Bauchi, Nigeria. The data collected was cleaned, normalized and organized in a form suitable for data mining process using WEKA platform. Table 1 shows the data format used for the research.

Table 1. Data format.

S/N	Variable Name	Variable Format	Variable Type
1.	Gender	Male, Female	Categorical
2.	Appointment Status (Appt. Status)	Permanent, Temporary, Contract	Categorical
3.	Employment Status (Emp. Status)	Old, New	Categorical
4.	Rank	Professor, Reader, SL,L1,AL, GA	Categorical
5.	Age	25, 30,…	Numerical
6.	University Working Experience (Univ. Exp.)	Year	Numerical
7.	Academic Qualification (Aca. Qual)	PhD, Master, Bachelor	Categorical
8.	Year of the Last Qualification	1996,1997	Numerical
9.	Professional Qualification (Prof. Qual.)	Yes, No	Categorical
10.	Average Unit Load	10, 15, 20, 24….	Numerical
11.	Formative Assessment Points	1, 2, 3….	Numerical
12.	Summative Assessment Point (SAP)	1, 2, 3….	Numerical
13.	Weighted Max Point (WmaxP)	20, 30, 40, 50……	Numerical
14.	Performance	Satisfactory, Average, Poor	Categorical

The data consists of both categorical and numerical data making it suitable to perform this experiment.

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 05, No. 01, April 2019.

3.2 The Existing Model (C4.5)

The C4.5 algorithm is an improvement of the ID3 algorithm, developed by Quinlan Ross in 1993. It uses gain ratio as an extension of gain information of ID3.

3.2.1 Mathematical Model (C4.5)

Let's consider the following probability distribution $(P = p_1, p_2, p_3, p_4, \dots, p_v)$ and a dataset D and define the information carried by the distribution otherwise known as the entropy of P, proposed by [14]-[15], [18] given as:

$$Entropy(P) = -\sum_{j=1}^{v} p_j \log_2(p_j) \tag{3}$$

And the gain information for a test A is given by:

$$Gain(P, A) = Entropy(P) - \sum_{j=1}^{v} ((p_j) Entropy(p_j)) \tag{4}$$

We can define the splitting information in the form:

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|_{D_j}|}{|_D|} \log_2 \left(\frac{|_{D_j}|}{|_D|} \right) \tag{5}$$

Let us consider a dataset D of some certain attributes with element $a_1, a_2, a_3, \dots a_n$, where the gain ratio of such data set is given by:

$$Gain_{Ratio(D,A)} = \frac{Enropy(D) \sum_{j=1}^{i} \left(p_j * Enropy(p_j) \right)}{Splitting_{Info}} \tag{6}$$

The two limitations associated with ID3; i.e., latency and over-fitting error are being improved by the gain ratio. The algorithm of C4.5 is shown below.

3.2.2 Algorithm of C4.5

Input: an attribute-valued dataset D
1. Tree = {}
2. If D is "pure" OR other stopping criteria met then
3. terminate
4. end if
5. for all attribute a ∈ D do
6. Compute the gain ratio if we split on **a**
7. end for
8. $a_{best} =$ Best attribute according to the above-computed criteria
9. Tree = Create a decision node that tests a_{best} in the root
10. D_v = Induced Sub-dataset from D based on a_{best}
11. for all $D_v$do
12. $Tree_v =$ C4.5 (D_v)
13. Attach $Tree_v$ to the corresponding branch of the Tree
14. end for
15. return Tree

3.3 The Proposed Model (EC4.5)

Suppose that we replace the split inform; i.e., denominator in Equation (6) with β

$$Gain_{Ratio(D,A)} = \frac{Enropy(D) \sum_{j=1}^{i} \left(p_j * Enropy(p_j) \right)}{\beta} \tag{7}$$

The gain ratio is known to present a better result than the information gain if the set of element $a_i \neq a_j$, but if $a_i = a_j$, the result of the gain ratio and information gain is the same. We can see that from (4) if $\beta = 1$:

$$GainRatio(D,A) \;\; = \; Enropy(D)\sum_{j=1}^{i}\left(p_j \; Enropy\left(p_j\right)\right) \tag{8}$$

Equation (7) shows that when split info (β) =1, then ID3 = C4.5.
To overcome this:
If we let β be the subject, we can rewrite (7) as:

$$\beta = \frac{Enropy(D)\sum_{j=1}^{i}\left(p_j * Enropy\left(p_j\right)\right)}{Gain\_Ratio(D,A)} \tag{9}$$

Now, from (7), for β =1

Consider a Taylor's series

$$1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots \frac{x^n}{n!} \tag{10}$$

For $x = 1$, the series can be rewritten as:

$$\frac{\beta}{1!} + \frac{\beta}{2!} + \frac{\beta}{3!} + \cdots \frac{\beta}{n!} \tag{11}$$

By taking the limit at $n \to \infty$,

$$\frac{\beta}{1!} + \frac{\beta}{2!} + \frac{\beta}{3!} + \cdots \frac{\beta}{n!} \quad n \to \infty \; = e^{\beta} \tag{12}$$

e^{β} is called optimal split information; therefore, it optimizes the splitting information by splitting the value away from (1). It works for both cases: when $a_i = a_j \; and \; when \; a_i \neq a_j$. The new technique suggests the introduction of a new parameter to the splitting information. We denote this $E - split$ and it is defined in the form:

$$E - split = e^{[SplitInfo_A(D)]} \tag{13}$$

This is equivalently defined as:

$$E - split \;\; = Exp\left(\sum_{j=1}^{v}\frac{\left|D_j\right|}{\left|D\right|}\log_2\left(\frac{\left|D_j\right|}{\left|D\right|}\right)\right) \tag{14}$$

The introduction of the new parameter suggests that the splitting values are spread around the value 1. This helps in obtaining a better result. The division of Equation (2) by Equation (5) leads to the new method $E.C4.5$ which it is defined as:

$$E.C4.5 = \left(\frac{Gain(P,A)}{E - split}\right) \tag{15}$$

3.3.1 Algorithm of the Proposed EC4.5

Input: an attribute-valued dataset D
1. Tree = {}
2. If D is "pure" OR other stopping criteria met then
3. terminate
4. end if
5. for all attribute a ∈ D do
6. Compute the gain ratio using exponential split if we split on **a**
7. end for
8 .a_{best} = Best attribute according to the above computed criteria
9. Tree = Create a decision node that tests a_{best} in the root
10.D_v = Induced Sub-dataset from D based on a_{best}
11. for all $D_v$do
12. $Tree_v$ = EC4.5 (D_v)

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 05, No. 01, April 2019.

13. Attach $Tree_v$ to the corresponding branch of the Tree
14. end for
15. return Tree

3.4 Evaluation

We consider the following terms in evaluating the performance of the proposed EC4.5.
(a) TN (True Negative) is the number of correct predictions that an instance is invalid.
(b) FP (False Positive) is the number of incorrect predictions that an instance is valid.
(c) FN (False Negative) is the number of incorrect prediction that an instance is invalid.
(d) TP (True Positive) is the number of correct predictions that an instance is valid.

Also, the following performance measure was used to test the performance of the proposed EC4.5. Accuracy is the proportion of the total number of predictions that were correct:

$$Accuracy(\%) = \frac{(TN+TP)}{(TN+FN+FP+TP)} \tag{16}$$

Precision is the proportion of the predicted valid instances that were correct:

$$Precision(\%) = \frac{TP}{(FP+TP)} \tag{17}$$

Recall is the proportion of the valid instances that were correctly identified:

$$Recall(\%) = \frac{TP}{(FN+TP)} \tag{18}$$

F-Measure is derived from precision and recall values:

$$F-Measure(\%) = \frac{(2 \times Recall \times precision)}{(Recall + Precision)} \tag{19}$$

The F-Measure is used because despite the Precision and Recall values are valid metrics in their own right, one of them can be optimized at the expense of the other. The F-Measure only produces a high result when Precision and Recall are both balanced and significant.

4. IMPLEMENTATION OF THE PROPOSED MODEL (EC4.5)

To see how the new model works, we consider the following example. Suppose that we want to compare the performances of ID3, C4.5 and EC4.5 to decide whether the time will be good to play basketball. A two-week data collection was used.

Table 2. Experimental dataset [19].

Day	Outlook	Temperature	Humidity	Play
1	Sun	Hot	High	No
2	Sun	Hot	High	No
3	Overcast	Hot	High	Yes
4	Rain	Sweet	High	Yes
5	Rain	Cold	Normal	Yes
6	Rain	Cold	Normal	No
7	Overcast	Cold	Normal	Yes
8	Sun	Sweet	High	No
9	Sun	Cold	Normal	Yes
10	Rain	Sweet	Normal	Yes
11	Sun	Sweet	Normal	Yes
12	Overcast	Sweet	High	Yes
13	Overcast	Hot	Normal	Yes
14	Rain	Sweet	High	No

The classification of the target is "Should we play basketball?" The answer can be either yes or no. The weather attributes which include outlook, temperature and humidity take the following values:

Outlook = {Sun, Overcast, Rain}
Temperature = {Hot, Sweet, Cold}
Humidity = {High, Normal}

So, using the three now sets: the information gain (ID3), the gain ratio (C4.5) and the E-gain ratio (EC4.5) are calculated for the outlook based on temperature and humidity as shown in the appendix.

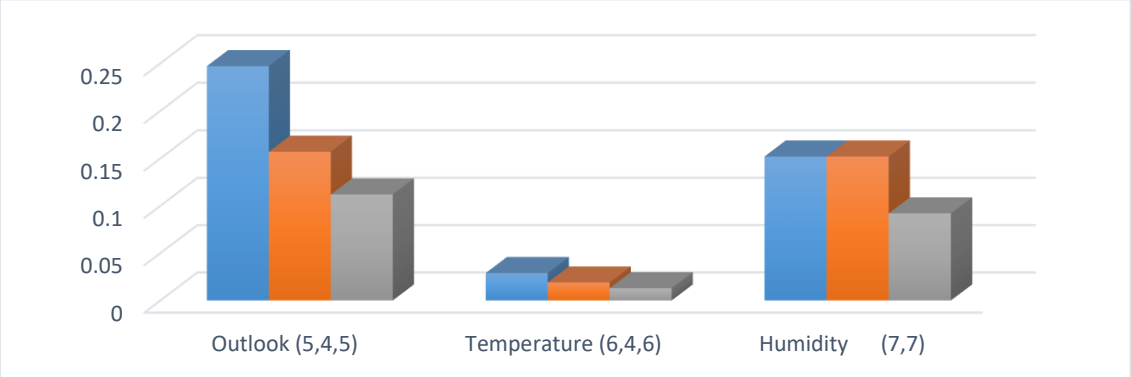


Figure 1. Outcome of the 3 classification algorithms.

From Figure 1, the three classification algorithms ID3, C4.5 and EC4 have the following outcome: outlook with 5,4,5 attributes shows that ID3 has a value of 0.247, C4.5 has a value of 0.157 and EC4.5 has a value of 0.112. Subsequently, temperature with 6,4,6 attributes shows that ID3 has a value of 0.029, C4.5 has a value of 0.019 and EC4.5 has a value of 0.013. However, humidity which has the same number of attributes of 7,7 leads to ID3 and C4.5 having the same value of 0.152. EC4.5 shows an improvement by having the value 0.092 which reduces the number of uncertainties in C4.5.

5. Results and Discussion

In the experiment, the values of the gain ratio (C4.5) and E-gain ratio (EC4.5) were first used to calculate the probability of uncertainty of some selected attributes with the highest instances. The outcome is shown in detail in Table 3. Figure 2 displays that EC4.5 is the optimal algorithm which has the lowest probability of uncertainty on all attributes and C4.5 has the highest probability of uncertainty.

Table 3. Probability of uncertainty outcome of gain ratio and E-gain ratio.

Selected Attribute	C4.5 (Gain Ratio)	EC4.5 (E-Gain Ratio)
SAP	0.931464	0.335235
Aca. Qual.	0.639899	0.227053
Rank	0.429688	0.114575
Univ. Exp.	0.148147	0.082564
Age	0.090222	0.030221
Prof. Qual.	0.310135	0.110069
Appt. Status	0.105773	0.02568
Gender	0.002881	0.001252
Emp. Status	0.000418	0.000313

Furthermore, C4.5 and EC4.5 classification algorithm, were trained and tested on the same dataset; the measures used for the algorithm performance evaluation were accuracy, precision, recall and F1 measure.

Table 4 illustrates the detailed results of the two classification algorithms.

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 05, No. 01, April 2019.

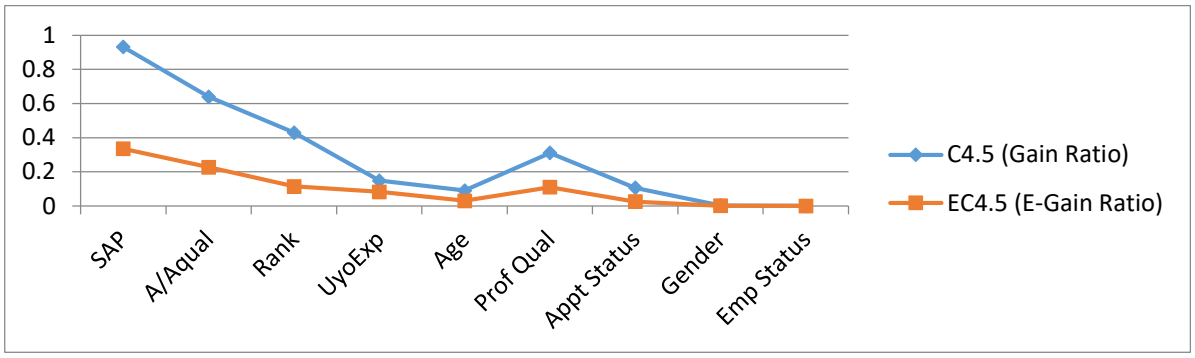


Figure 2. Probability of uncertainty outcome of gain ratio and E-gain ratio.

Table 4. Detailed classification accuracy results of C4.5 and EC4.5.

Performance Metrics	**C4.5**	**EC4.5**
Accuracy	51.27%	99.40%
Error Rate	48.73%	0.60%
Precision	0.513	0.994
Recall	0.513	0.994
F1 Measure	0.678	0.994

The detailed classification accuracies suggest that EC4.5 outperformed C4.5, because it has a lower FP rate of 0.003 and a TP rate of 0.994 which was used to calculate the accuracy using the performance metrics. Thus, EC4.5 is the optimal model of classification algorithm in this paper.

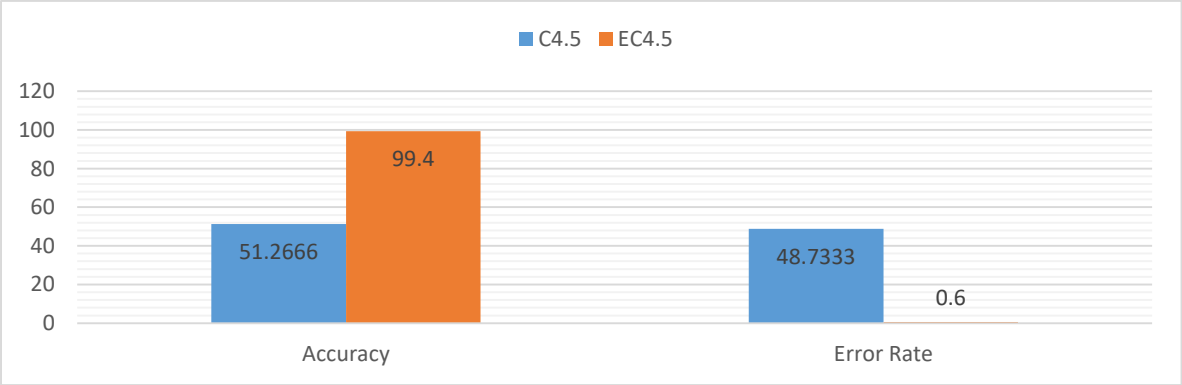


Figure 3. Accuracy and error rate of C4.5 and EC4.5.

Figure 3 shows that EC4.5 has the highest accuracy of 99.40% with an error rate of 0.60%, while C4.5 has an accuracy of 51.27% with an error rate of 48.73%.

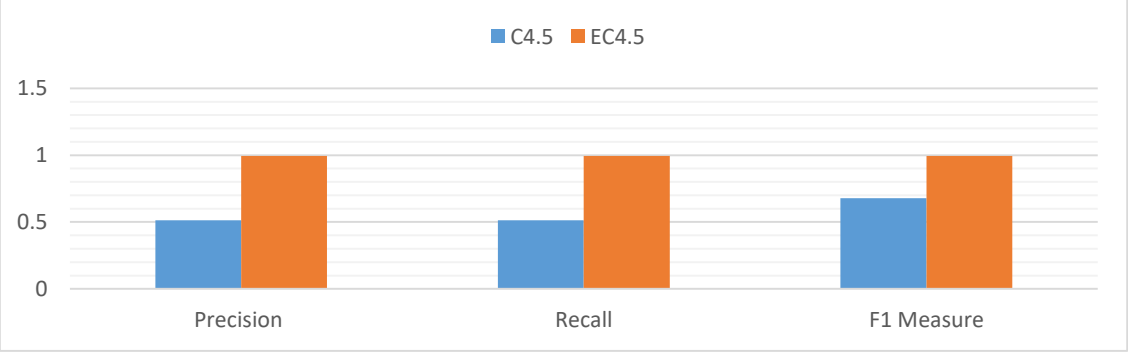


Figure 4. Precision, recall and F1 measure of C4.5 and EC4.5.

Figure 4 shows the detailed results of the compared algorithms, with C4.5 having the highest value over

EC4.5; under precision C4.5 has 0.513 and EC4.5 has 0.994; under recall C4.5 has 0.513 and EC4.5 has 0.994; and lastly under F1 measure C4.5 has 0.678 and EC4.5 has 0.994. The overall result suggested that EC4.5 is the optimal algorithm compared to C4.5.

6. CONCLUSIONS

This paper proposed a modified model (EC4.5). The proposed modification offers solutions to the limitations associated with C4.5 in terms of presenting an equivalent result with ID3 when the same number of attributes is used. After testing the two classifiers (C4.5 and EC4.5), the result of the experiment shows that EC4.5 outperformed, with an accuracy of 99.40%, whereas C4.5 has an accuracy of 51.27%. Based on the result of this research, EC4.5 was selected as the optimal algorithm. Future work should consider a hybrid approach to handle multi-dimensional data with large intervals using EC4.5 algorithm.

ACKNOWLEDGEMENTS

The authors would like to thank the management of Bayero University, Kano, Nigeria for their moral encouragement and financial support.

REFERENCES

[1] D. Clayson and M. J. Sheffet, "Personality and the Student Evaluation of Teaching," Journal of Marketing Education, vol. 2, no. 28, pp. 149-160, 2006.

[2] B. Hussina, A. Merbouha and H. Ezzikouri, "A Comparative Study of Decision Tree ID3 and C4.5," International Journal of Advanced Computer Science, vol. 3, no. 1, pp. 13-19, 2014.

[3] C. Romero, L. O. Juan and V. Sebastian, "A Meta-learning Approach for Recommending a Subset of White-box Classification Algorithms for Moodle Datasets," Journal of Theoretical and Applied Information Technology, vol. 6, no. 5, pp. 268-271, 2013.

[4] K. Gaganjot and C. Amit, "Improved J48 Classification Algorithm for the Prediction of Diabetes," International Journal of Computer Application, vol. 98, no. 5, pp. 13-17, 2014.

[5] M. M. Mazid and A. Shawkat, "Improved C4.5 Algorithm for Rule-based Classification," Proceedings of the 9th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases, pp. 296-301, 2010.

[6] A. Moustafa, E. Shahira and M. Essam, "Defining Difficult Laryngoscpy Findings by Using Multiple Parameters, A Machine Learning Approach," Egyptian Journal of Anaesthesia, vol. 33, no. 2, pp. 153-158, 2017.

[7] Y. Yuan and M. Shaw, "Fuzzy Sets and Systems," Elservier, vol. 69, no. 2, pp. 125-139, 1995.

[8] A. B. Adeyemo and O. Oriola, "Personal Audit Using a Forensic Mining Technique," International Journal of Computer Science, vol. 7, no.7, pp. 222-231, 2010.

[9] K. Asha, A. M. Gowda and M. Jayaram, "Comparative Study of Attribute Selection Using Gain Ratio and Correlation-based Feature Selection," International Journal of Information Technology and Knowledge Management, vol. 2, no. 2, pp. 271-277, 2010.

[10] G. Chaitn, "Algorithmic Information Theory," Journal of Research and Development, vol. 8, no. 4, pp. 350-359, 2000.

[11] S. Hardikar, A. Shrivastava and V. Choudary, "Comparison between ID3 and C4.5," International Journal of Computer Science, vol. 2, no. 7, pp. 34-39, 2012.

[12] L. Gaurav and G. Hiaesh, "Optimization of C4.5 Decision Tree Algorithm for Data Mining Application," International Journal of Information Technology and Knowledge Management, vol. 3, no. 3, pp. 2250-2459, 2013.

[13] R. Hartley, "The Function of Phase Difference in the Binaural Location of Pure Tones," Journal of Advanced and Applied Sciences, vol. 13, no. 6, pp. 373-385, 2000.

[14] S. Kumar and E. Ramaraj, "Modified C4.5 Algorithm with Improved Information Entropy and Gain Ratio," International Journal of Engineering Research and Technology, vol. 2, no. 9, pp. 2768-2773, 2013.

[15] K. Santhosh, "Modified C4.5 Algorithm with Improved Information Entropy," International Journal of

Engineering Research & Technology, vol. 2, no. 14, pp. 485-512, 2013.

[16] W. Yisen, S. Chaobing and X. Shu-Tao, "Improving Decision Trees by Tsallis Entropy Information Metric Method," Proc. International Joint Conference on Neural Networks, Vancouver, BC, Canada, IEEE Xplore, 24 – 27, July, 2018.

[17] M. M. Mazid, "Improved C.4.5 Algorithm for Rule-based Classification," in Mastorakis Nikos (ed.), Artificial Intelligence Knowledge Engineering and Database, vol. 7, no. 5, pp. 296-301, 2017.

[18] A. Neeraj, G. Bhargava and M. Manish, "Decision Tree Analysis on J48 Algorithm for Data Mining," International Journal of Advanced Research in Computer Science, vol. 3, no. 6, pp. 22-45, 2013.

[19] M. Dragon and G. Lujbisa, "The Use of Data Mining for Basketball Matches Outcome Prediction," Proc. of the 8th IEEE International Symposium on Intelligent Systems and Informatics, pp. 309-312, Serbia, 2010.

[20] I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes and S. J. Cunningham, "Weka: Practical Machine Learning Tools and Techniques with Java Implementations," (Working Paper 99/11), Department of Computer Science, University of Waikato, Hamilton, New Zealand, vol. 31, pp. 76-81, 2000.

[21] I. Al-Turaiki, M. Alshahrani and T. Almutairi, "Building Predictive Models for MERS-COV Infectionss Using Data Mining Techniques," Journal of Infection and Public Health, vol. 9, no. 6, pp. 744-748, 2016.

[22] L. Yi-bin, W. Ying-ying and R. Xue-wen, "Improvement of ID3 Algorithm Based on Simplified Information Entropy and Coordination Degree," Proc. of Chinese Automation Conference , IEEE Xplore, Jinan, China , vol. 1, no. 3, pp. 88-92, 2017.

[23] G. Attilio and N. Filipo, "Search-Intensive Concept Induction," International Journal of Computer Science, vol. 7, no. 6, pp. 137-145, 2000.

[24] T. G. Kumar, "Advanced Applications of Neural Networks and Artificial Intelligence," International Journal of Information Technology, vol. 2, no. 6, pp. 57-68, 2012.

[25] S. Mardikyan and B. Badur, "Analyzing Teaching Performance of Instructors Using Data Mining," Journal of Informatics in Education, vol. 10, no. 2, pp. 245-257, 2011.

APPENDIX

Result of the three classification algorithms

Outlook			Temperature			Humidity		
Algorithms	Attributes	Calculated Values	Algorithms	Attributes	Calculated Values	Algorithms	Attributes	Calculated Values
ID3	5,4,5	0.247	ID3	6,4,6	0.029	ID3	7,7	0.152
C4.5		0.157	C4.5		0.019	C4.5		0.152
EC4.5		0.112	EC4.5		0.013	EC4.5		0.092

ملخص البحث:

تعد الخوارزمية (C4.5) من أكثر خوارزميات التصنيف شيوعاً. وعلى الرغم من أفضليتها على الخوارزمية (ID3)، فهي تعطي نتيجة مماثلة لتلك التي يتم الحصول عليها باستخدام الخوارزمية (ID3)، وبخاصة عند استخدام العدد نفسه من السمات. وتقترح هذه الورقة تقنية تعالج هذا القصور في الخوارزمية (C4.5). وقد تم تقييم أداء التقنية المقترحة بناءً على الدقة.

يستخدم الباحثان معلومات الفصل الأُسي (EC4.5) للاستفادة من السمة المركزية لمجموعة البيانات نفسها، ويقدمان نموذجاً محسَّناً لخوارزمية التصنيف (C4.5) بناءً على سلسلة تايلور. وقد أعطى النموذج المحسّن المقترح نيجة أفضل بكثير مقارنة بنتائج الخوارزمية التقليدية (C4.5) تدلّ عليها الدقة الأعلى للنموذج المقترح والبالغة 99.4% مقارنة بدقة الخوارزمية الأصلية (C4.5) البالغة 51.27%.

Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 05, No. 01, April 2019.

SENTIMENT ANALYSIS OF ELECTRONIC PRODUCT TWEETS USING BIG DATA FRAMEWORK

Sunil Kumar, Vartika Koolwal and Krishna Kumar Mohbey

(Received: 8-Jan.-2019, Revised: 27-Feb.-2019, Accepted: 13-Mar.-2019)

ABSTRACT

Nowadays, social media has become more popular due to the advancement of Internet technologies and smartphone devices. Such platforms have generated interest among users to give their opinion. Social media-like Twitter- also plays an important role for business companies. Based on customer opinion about any product, business companies came to know more about customer choices. In the current scenario, millions of tweets are generated by people every year. But handling these huge unstructured tweets is not possible through the traditional platform. Therefore, big data framework, such as Hadoop and Spark, is used to handle such kind of large data.

In this paper, different sale tweets are used to analyze the sentiments of customers regarding electronic products. The experimental results of the proposed work will be useful for various business companies to take business decisions, which will further enhance the product sales.

KEYWORDS

Twitter, Spark, Big data, Flume, Sentiment analysis.

1. INTRODUCTION

Social media platforms, such as Twitter, Facebook and Instagram, have become vital constituents of daily life. People use these media to express their feelings, opinions, expressions, views and experiences about places or things [1]. Sentiment analysis is used to classify public opinion towards a particular topic or product. Various prominent categories of sentiment analysis, such as machine-learning [2], lexicon-based [3] and hybrid [4] categories, are worked upon. A progressive practice has grown to draw out the information from data available on social networks. This data has huge potential and can be harnessed for business-driven application [5], such as movie review [6], product advertisement, public election [9], brand endorsement and many more.

For real-time data analysis, Twitter is the rational choice due to a large amount of relevant data, compact and concise tweets up to 280 characters and simplicity to post an opinion. Real-time tweets are collected using hashtags (like #iphone, #OppoF9Pro). Opinion mining [7] approach was used to find polarity of tweets such as positive, negative and neutral. Knowing the collective sentimental affinity could help companies transform their strategies [5].

For many years, the problem of sentiment analysis has been studied and proposed solutions suffer from certain disadvantages. Constant problems with these approaches were centralized environment and time-consuming techniques, which scare many computational resources [8]. Furthermore, these standard approaches work on limited tweets and are not able to handle large size of tweets. Dubey et al. [9] proposed opinion-lexical approach in R platform to get insight about public opinion on political diplomats. However, the proposed approach works on a small dataset of approx. 3000 tweets. So, for enhancing the capability to handle a large number of tweets, we require distributed or parallel processing techniques, such as Spark.

Al-Saqqa et al. [10] collected 4 million Amazon customers' review dataset for large-scale sentiment analysis under Apache Spark framework. The dataset was tested for supervised machine-learning algorithm, where the model was trained using labeled training set. It applied classification techniques, where support vector outperforms Naïve Bayes and logistic regression, attaining an accuracy of 86%.

S. Kumar[1], V. Koolwal[2] and K. K. Mohbey[3] are with Department of Computer Science, Central University of Rajasthan, Ajmer, India.
Emails: [1]sunil.cs@gmail.com, [2]vartikakoolwal14@gmail.com and [3]kmobhbey@gmail.com

In the age of Internet with such massive data, there is a need for faster computing and distributed storage, leading to a framework like Apache Spark, Apache Hadoop and Map Reduce techniques. Spark has emerged as the most popular big data processing engine. It improves over its predecessor, i.e., Hadoop MapReduce. MapReduce provides a simple model for writing programs that could execute in parallel in cluster. Spark improves MapReduce in three ways. Firstly, Spark engine can execute more general Directed Acyclic Graph (DAG) of operators than the rigid map-then-reduce format of MapReduce. Secondly, it has a rich set of transformation, which enables the output of one operation directly fed into another operation. Lastly, Spark extends with in-memory processing. Developers can instruct to cache any point in a processing pipeline, so future operations that need same data don't require to reload or recompute. It can be launched as a stand-alone or on cluster modes like Hadoop YARN, Apache Mesos and Kubernetes. It can integrate with distributed storage, such as HDFS, HBase and Cassandra. It is fast, much easy to use because of its high-level APIs in Java, Scala, Python and R. It has libraries, like MLlib for machine learning in Big data, GraphX for graph processing, Spark SQL and Spark Streaming [11].

In this paper, we do not propose any sentiment-prediction technique, but our aim is to analyze the eminent techniques regarding electronic products. We aim to perform sentiment analysis of data collected from Twitter using flume. These tweets are classified based on supervised learning approaches, such as Naive Bayes, SVM, Decision Tree, Random Forest and Logistic Regression classifier.

The remainder of the paper is arranged in the following manner. Section 2 represents related work. Section 3 is regarding big data processing using MapReduce, Spark and MLlib. Classification techniques are shown in section 4. In section 5, we present the sentiment analysis framework. Moreover, section 6 demonstrates the comprehensive experimental results. Conclusion and future work are presented in section 7.

2. RELATED WORK

Semantic analysis is the investigation of people's opinions, beliefs, attitudes and emotions towards an entity, such as products, services, events, issues and topics [1]. It is the field of machine learning which has gained the attention of researchers since the beginning of the century. Miller et al. [12] introduces WordNet, an online database for English language semantic processing using synonym sets (synsets) relationship. SentiWordNet [13] is an advancement of WordNet as a tool for knowledge-based word level processing *via* building a dictionary to find a score of each word.

Kim and Hovy [16] operated on a word granularity by using initially some seed words and using them to create a net; they proceeded further to sentence level by combining the strengths of the words, as they classify people's opinions. Moreover, Wilson et al. [17] operated on a phrase level, by running a supervised learning approach to determine the polarity or neutrality of phrases. Furthermore, document granularity [18] used word frequency and part of speech approach on Amazon reviews in categories, like books, DVDs, electronic and kitchen appliances to evaluate the response of people about the products.

Twitter streaming API[1] was used to gather data for product sentiment analysis [3]. The aim of using twitter data is to understand public opinion. Around 60,000 tweets were collected using Twitter API to analyze customer opinions on widely used smartphones in Korea [21]. Kumar et al. excavated opinions of the people about the quality of services provided by Airtel company [22]. For this purpose, they collected 80,000 tweets using the hashtag "#Airtel". They assessed them using Naïve Bayes approach with an accuracy of 80.9% on Mahout installed over Hadoop to classify them into different classes. They used term frequency and inverse document frequency for internal processing.

Various techniques, such as machine learning [2], entropy-based [24] and tree-kernel [25] techniques, are used for Twitter sentiment analysis. The hybrid algorithms presented in [26] for Twitter feed classification improve accuracy when compared with similar techniques. To increase accuracy, word sequence disambiguation [15] and negation handling [16] could be used. In [27], the authors mined tweets with emoticons and punctuations. They concluded that Naïve Bayes performance and accuracy

[1]Twitter Apps. Available online: http://www.tweepy.org/

are higher than those of SVM. Emoticons and hashtags [28] are employed as sentiment labels to carry out KNN classification of diverse sentiment types. Kaur et al. [28] have used Spark for processing large data. They have also used Bloom filter for inspecting element membership in any proposed set and space compaction.

Agarwal et al. [25] used unigram model to classify Twitter data into 4 classes: positive, negative, neutral and junk, where junk included tweets not understood by a human annotator. They investigated on tree kernel and feature-based models and reported that these models outperform the unigram baseline. They highlighted that for feature analysis, prominent features were a combination of the prior polarity of words and their parts-of-speech-tags. However, they used manually annotated Twitter data for the test.

Kaptein [29] studied what influence the tweets have on the reputation of the company. They explored the sentimental-bearing text (i.e. subjective text) for factual information to derive reputational polarity. For example, *Nokia Smartphone blasted while charging* has a negative reputation for Nokia Company. They suggested that developing a polarity lexicon for the specific domain will be cost-beneficial.

In [10], the authors retrieved 4 million tweets, which required bulk processing speed and distributed storage, signifying the need for Big Data frameworks, like Hadoop and Spark. These frameworks are required to meet up the shooting data generation demand. Many researchers are using similar frameworks for tweet analysis [30]. Baltas et al. [31] has used Twitter data with Spark platform. In the proposed approach, they have used binary and ternary classification. The result of F-measure of feature vector of logistic regression indicated 62.8% positive, 59.2% negative and 54.2% neutral. Chan and Thein [32] used sentiment analysis on 60k real-time tweets using Apache Flume on iphone mobile product. The results show that linear SVM performs better than NB by 10 % and better than logistic regression by 2%.

Earlier studies have shown that the traditional approach is suitable for limited data only. But, if we have a large amount of real-time tweets, we can't process them with normal architecture and traditional approaches. Therefore, it is high time to develop a framework with distributed processing to improve accuracy and performance of the models. So, in this paper, we are working with Spark framework and have used Flume for fast data retrieval. We have demonstrated the results of semantic analyzers and their machine learning validation is shown in tabular formats and graphs to render a complete picture about accuracy gained. We have not formulated any semantic prediction technique, but have analyzed SVM, NB, logistic regression, decision tree and random forest techniques on unstructured real-time electronic product tweets using Big Data framework. We have attained the average accuracy of 91% in logistic regression that is outperforming all the competing techniques.

3. BIG DATA PROCESSING

Big data deals with large datasets which require complex processing and need huge storage. Big data frameworks are listed below.

3.1 Hadoop

Hadoop software library is an open source implementation of the MapReduce framework. It enables distributed and parallel processing of large datasets. It also provides distributed storage on cluster of computers [33]. Hadoop core contains MapReduce and Hadoop Distributed File System (HDFS). HDFS is responsible for storing large datasets on the cluster, which are partitioned into blocks and distributed into nodes.

3.2 MapReduce

MapReduce model allows distributed processing across multiple nodes in a cluster. It contains a map and a reduce function procedure, called mapper and reducer, respectively [34]. Input data is partitioned into the mapper phase and transferred to workers to execute the map function; each worker output is in key-value pairs after processing the data. Shuffle phase sorts the output and groups it by key. Reducer calls for every unique key and gets a set of values associated with key. MapReduce framework deals with the underlying parallelization, adjustment to internal failure, information

distribution between nodes and load adjustment. Data is replicated and distributed across nodes to improve both accessibility and reliability.

3.3 Spark Framework

Apache Spark[2] is a fast and general framework for large-scale data processing. It is the improvement of Hadoop framework. Hadoop is ideal for large batch processing when we require to go through all data. However, its performance drops quickly for certain scenarios, e.g. when we have to deal with graph-based or iterative algorithms. Hadoop does not cache intermediate results but instead, it flushes the data to the dish in between each step. In contrast, Spark has a Directed Acyclic Graph (DAG) execution engine that allows cyclic data flow and in-memory computing. So, it can execute programs up to 100x times faster than Hadoop. It contains a set of libraries which combines streaming, SQL, graph processing and machine learning in a single engine. It provides many high-level APIs in Python, Scala, java and R and can run on Hadoop or standalone while using different data sources, such as, HDFS, Cassandra or HBase. It provides a programming model that hides the partitioning of dataset in cluster, using a new data structure called Resilient Distributed Dataset (RDD) [35]. RDD is an immutable distributed collection of records partitioned into different nodes of the cluster. Data-sharing abstraction property of RDD allows to run a wide range of APIs provided by Spark: MLlib, Spark streaming, Spark SQL and GraphX (graph processing). By default, RDDs are short-lived, so if they are used in an action, they need to be recomputed. However, they can persist in memory for frequent reuse.

3.4 MLlib

MLlib is Spark's largest distributed learning library. It includes fast, scalable and easy implementation of common learning algorithms of machine learning, including classification, regression, clustering and collaborative filtering [36]. The library also has low-level primitives for convex optimization, statistical analysis tools, distributed linear algebra and feature extraction and provides various I/O formats, such as LIBSVM format, Spark SQL data integration[3] and MLlib's internal format. It shows excellent performance and scalability to handle larger problems.

4. CLASSIFICATION TECHNIQUES

This section describes sentiment analysis phases. The complete process of sentiment analysis is shown in Figure 1. The following supervised classification approaches are used to predict the polarity of a tweet.

4.1 Naïve Bayes

Naïve Bayes is an easy probabilistic classifier, which uses Bayes Theorem with an assumption of high (naïve) independence between features. It had proven effective in many application domains, like system performance management [37], text classification, medical diagnosis and many more. It assigns the most favourable class to a given instance according to its feature vector which is given by:

$$P(CL \mid X) = \frac{P(CL) * P(X \mid CL)}{P(X)} \quad (1)$$

where, $X = (x_1, x_2, \ldots, x_n)$, indicating some independent feature vectors.
CL : L possible outcomes (classes).
X : Tweet needing to be classified.
P (CL | X): Posterior probability.
P(CL) and P(X) : Prior probabilities.

4.2 Support Vector Machine

Support Vector Machine carries out classification by searching for the hyperplane (boundary dividing

[2] Spark https://spark.apache.org
[3] Spark SQL https://spark.apache.org/sql/

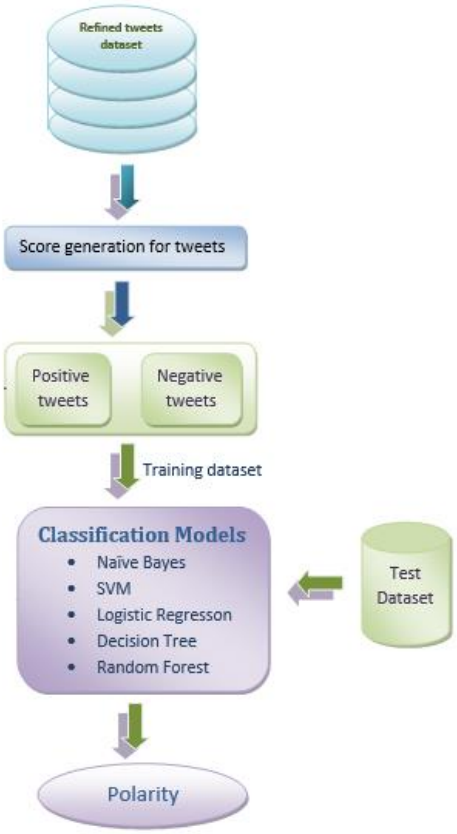


Figure 1. Sentiment analysis of tweet dataset.

one entity set from another) that maximizes the margin between two classes. Hyperplanes are explored using "important training tuples" (support vectors) along with margins [38]. SVM can be implemented on both linear and non-linear datasets. SVM as a supervised learning classifier is popular due to its high reliability, varied application usage and less vulnerability to overfitted model [39].

We traverse linearly separable class using two-class problems. We are given a dataset S as (P1, Q1), (P2, Q2),……..(P|S|, Q|S|), where Qj is the class label whose value is from +1 to -1(Qj ∈ (-1, +1)). Qj is associated with Pj set of training tuples.

Any hyperplane can be defined as P set of points satisfying

$$W.P - B = 0 \tag{2}$$

where, W is normal vector to the hyperplane. $\frac{B}{||W||}$ is the offset of the plane from the origin and normal vector W.

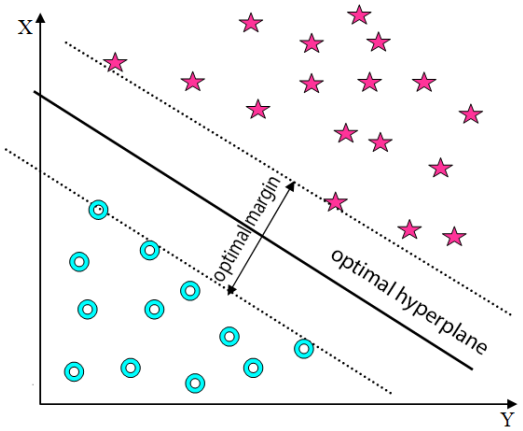


Figure 2. Support vector machine classifier.

We can plot multiple separating lines. We have to find the "best" line (least classification error), in general, best "hyperplane" by the maximum distance of the hyperplane to the closest negative instance and positive instance. Figure 2 shows SVM optimal hyperplane in training with sample tweets to classify positive tweets (star-shaped) and negative tweets (disk-shaped).

4.3 Decision Tree

Decision Tree is a flow-chart like structure, where each non-leaf node signifies test condition on the attribute; branches indicate the result of test and leaf node represents class label of entity set. First and topmost node is root node [25]. Tree is explored from top to bottom indicating classification rules. It is a decision support tool which is used to display the outcome of test condition, resource cost, utility along with an algorithm that contains a statement of conditional control.

Decision tree can be converted into decision rules by association rules with target variable on right-hand side. A decision tree can be used in temporal or causal relations [40]. Figure 3 shows decision tree classification processing based on test condition.

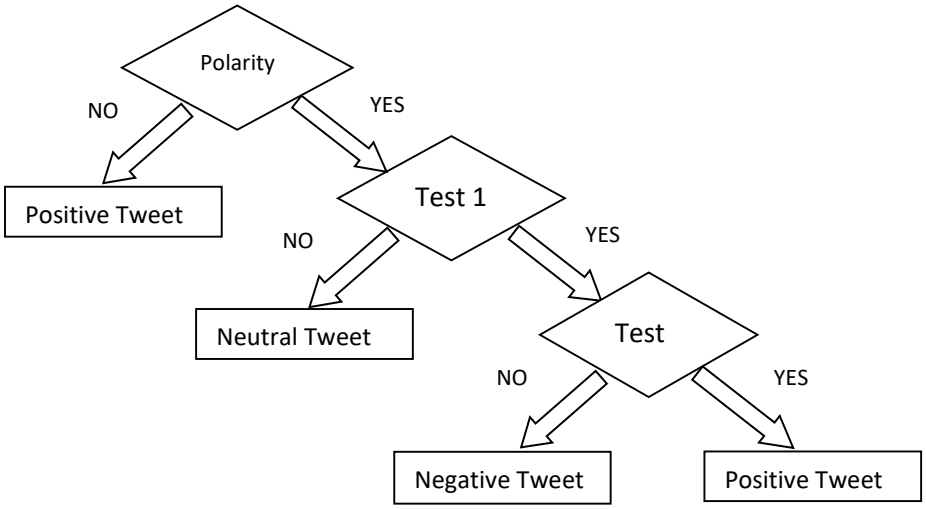


Figure 3. Decision tree classifier.

4.4 Random Forest

Random forest classifier is a tree classifier which is generated using independently selected random vector from input dataset. Each tree for most favourable class casts one vote to classify input vector [41]. It uses one or more combinations of features at every node to expand a tree. Bagging is a method to make training set *via* randomly drawing N replacement examples (N is the size of original training set used for feature selection). Every input instance can be classified by exploring most desirable voted class by all forest trees. We can use GINI index as a measure of attribute selection, which weights attribute impurity of all classes. For a given training dataset D, choosing one cast and ascertaining that it belongs to a class C_i, could be written as:

$$\sum \sum_{j \neq i} \left(\frac{f(c_i, D)}{|D|} \right) \left(\frac{f(c_j, D)}{|D|} \right) \tag{3}$$

where, $\frac{f(c_i,D)}{|D|}$ is the probability of that labelled class belongs to class C_i.

4.5 Logistic Regression

Logistic regression is a predictive classifier that is used to a model-dependent variable using logistic function. Dependent variable is a categorical value having two categories labelled as "0" and "1" like (loose or win, sick or not sick, true or false, tea or coffee). Independent variable is numerical or categorical value. It is used to classify observations, in terms of whether an observation belongs to a particular category or not (positive tweet or negative tweet in our problem).

Types of Logistic Regression:

- Binary Logistic Regression: models binary outcome (yes/no).
- Ordinal Logistic Regression: models an ordered response (completely disagree, disagree, somewhat agree, agree).
- Nominal Logistic Regression: models a multilevel outcome which is insensitive to ordering (choice of a transport mode such as bus, car, train).

Logit (log-odds) is a function which is equivalent to log odds of variables. If p is a probability of occurrence of an event (E= 1), then $\frac{p}{1-p}$ represents the corresponding odds. Logit (E) is given by:

$$logit(E) = log\left(\frac{p}{1-p}\right) \tag{4}$$

A logistic curve is obtained by a logistic function. Logistic curve is just like a sigmoid curve the input of which is as any real value k (k € R), while the output value falls between (0, 1). Logistic curve is shown in Figure 4. Logistic function (k) is given by:

$$p(k) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 k)}} \tag{5}$$

where, p (k) is the probability of dependent variable.
β_0 : intercept from the linear regression equation.
$\beta_1 k$: Regression coefficient multiplied by some predictor value.
e : Base e indicates the exponential function.

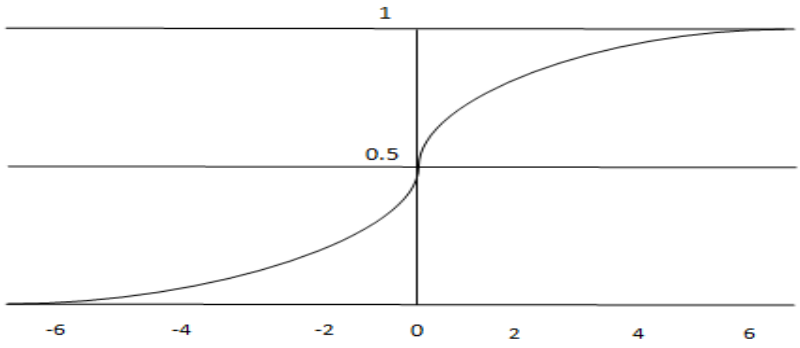


Figure 4. Logistic regression.

5. SENTIMENT ANALYSIS FRAMEWORK

We present a framework for sentiment analysis which includes data collection, pre-processing, sentiment score calculation for tweets, classification and polarity prediction.

5.1 Data Collection Using Twitter API by Flume

Twitter is a corpus of 500 million published tweets by 321 million active monthly users[4]. This real-time data provides immense opportunities to study social trends. Crawling data from Twitter was collected using Flume. Flume links Flume agent with web servers. This is done with API keys extracted from Twitter developer's account. Twitter delivers Rest API and Streaming API to different client systems to absorb tweets. Figure 5 shows the process of data retrieval using Flume agent. Tweets are collected from source to channel and then from channel to HDFS sink. Different hashtags are used to collect live-stream data from Twitter. Description of used hashtags and collected tweets is shown in Table 1.

[4] Statista 2019, February 2019, Number of monthly active Twitter users worldwide from 1st quarter 2010 to 4th quarter 2018 (in millions). Available: https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

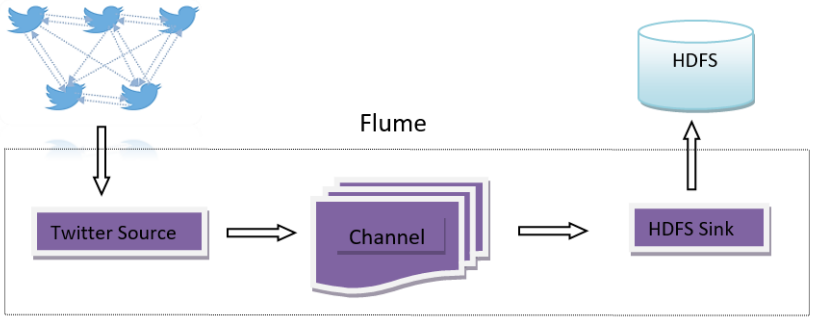


Figure 5. Twitter data collection.

Table 1. Hashtag description.

CATEGORY	HASHTAGS/KEYWORDS	# TWEETS
TWEETS FOR MOBILE PHONES	#Samsung # vivo #iPhone #htc #OppoF9Pro #Samsung # GooglePixel3XL #iPhone #htc #MiNote4 #motoG	1,00,000
TWEETS FOR LAPTOPS	#MacBookPro # iMac #HpEliteBook #ThinkPadLenovo #MSIGaming #chromebook # DellXPS #HPEnvy #AcerSwitch	70,000
TWEETS FOR TELEVISION	#SonyBraviaKLV # AndriodTv #SamsungQLED #TCL #LGLED #PanasonicSmartTv # VizioLcd #rokuTv #OLEDTV	50,000

Data extracted from Twitter using Twitter API comes in JSON format. Figure 6 is a snapshot of raw tweets in JSON format. However, JSON structure is not understood by user completely. Therefore, JSON Validator was used to validate data into a particular structure. Figure 7 shows the refined structural tweets after processing raw tweets in JSON format.


```
                        },
                        "medium": {
                                "w": 480,
                                "h": 360,
                                "resize": "fit"
                        }
                },
                "id_str": "1076327578963591170",
                "expanded_url": "https://twitter.com/whitestone_UK/status/1076327565764753782O/photo/1",
                "media_url_https": "https://pbs.twimg.com/media/D0UnR_GW0AACrBM.jpg",
                "id": 1076327578963591170,
                "type": "photo",
                "media_url": "http://pbs.twimg.com/media/D0UnR_GW0AACrBM.jpg",
                "url": "https://t.co/nPvSiu1ftc"
        }]
},
"entities": {
        "urls": [{
                "display_url": "buff.ly/2UUJwtK",
                "indices": [210, 233],
                "expanded_url": "https://buff.ly/2UUJwtK",
                "url": "https://t.co/a3IOQVXJ8t"
        }],
        "hashtags": [{
                "indices": [71, 87],
                "text": "Upgrade"
        }, {
                "indices": [88, 95],
                "text": "Update "
        }, {
                "indices": [96, 104],
                "text": "acer"
        }, {
                "indices": [105, 112],
                "text": "liquid"
```


Figure 6. Sample of raw tweets in JSON format collected from Twitter.

5.2 Pre-processing of Tweets

One of the major tasks of semantic analysis is data filtering. It helps improve the efficiency of the classifier. Following are the pre-processing steps:

id	date	time	favorited	retweet	text
1.06E+18	11/17/2018	8:13:38	FALSE	1	iPhone XR review: Bright colors, best value - Six Colors https://t.co/BAzA9spXWq iPhone XR Review #cellular... https://t.co/ytftiNJ6PL
1.06E+18	11/17/2018	8:13:30	FALSE	0	Full digest if #Apple news from @AppleLoop: New #iPhone #Problems, #MacBook #SecurityConcerns, #iPhoneSales Force R... https://t.co/TAWtdToyg
1.06E+18	11/17/2018	8:11:09	FALSE	0	Cacoteo Radio No App Needed To Listen visit https://t.co/Wvn56oxvbo and hit play #Iphone #Android Listen Anywhere N... https://t.co/s7DSY15pQB
1.06E+18	11/17/2018	8:10:58	FALSE	0	#SWEEPSTAKES! Win #iPhoneXS, AirPods, iPhone cable and VideoProc license code to process your iPhone videos! Try yo... https://t.co/thS4z9c9P9
1.06E+18	11/17/2018	8:10:09	FALSE	0	i praise only u, u are my jesus. so y am i the one being crucified #DeepBiblicalImagery #poetry #sosaad #iphone #ipod #gangnamstyle
1.06E+18	11/17/2018	8:09:37	FALSE	0	Cambridge Sunrise #cambridgebyphoto #cambridgeshire #goldenhour #iphone #mobilography #cambridge #cambridgeuk... https://t.co/S7VuTmrYIb
1.06E+18	11/17/2018	8:09:15	FALSE	0	Check out Tweet Garage: a handy app that tweets for you while you sleep or do something else! Save up to 25 tweets!... https://t.co/amTg8jchZ2
1.06E+18	11/17/2018	8:08:28	FALSE	0	I migliori speaker AirPlay 2 https://t.co/hL1IkaUPJk #pcexpander #cybernews #apple #newsapple .@apple #iphone... https://t.co/r0ticHlzya
1.06E+18	11/17/2018	8:06:31	FALSE	0	8 BALL POOL BLOG #237("4 DOUBLE POTS IN A ROW")#8BallPool #8ball #gamestagram #iphone #blog #Gaming #miniclip... https://t.co/7Eal8Oe7XS
1.06E+18	11/17/2018	8:04:35	FALSE	0	#manga #onepiece #anime #iPhone Onepiece wallpaper UPDATE! Ver4!! https://t.co/NY7tcKPv9X with 2013 calendar!! https://t.co/YVi9UHi6Wk
1.06E+18	11/17/2018	8:04:10	FALSE	0	#Iphone Mobile Phone Insurance, #Ipad & Gadget Insurance from £1.99 a month click> https://t.co/exlWArfNc2 #darlobiz
1.06E+18	11/17/2018	8:02:17	FALSE	0	#Camera #SmartPhone #Android 6.0 IPS Full Screen 1GB+4GB WiFi Bluetooth GPS 3G GSM/WCDMA Backup Call #Mobile #Phone... https://t.co/dIgEGH
1.06E+18	11/17/2018	8:00:54	FALSE	0	Cambridge Sunrise #cambridgebyphoto #cambridgeshire #goldenhour #iphone #mobilography #cambridge #cambridgeuk... https://t.co/j52QD8m1iw
1.06E+18	11/17/2018	7:58:39	FALSE	0	Constantly amazed at the photo quality from an #iphone. Bee on Harakeke flower today https://t.co/Y04ZC10yNu
1.06E+18	11/17/2018	7:56:43	FALSE	0	A very German Chinese restaurant. #snapshot #german_restaurant #beijing #beer #paulaner #chinese #china #iphone... https://t.co/B4dfmlR8QI
1.06E+18	11/17/2018	7:46:03	FALSE	0	Hello #iphonex https://t.co/0rU6fjo4Rh
1.06E+18	11/17/2018	7:45:03	FALSE	0	Welcome to #MacTwo, your one-stop shop for quality #used #devices. Come see the latest #SecondHand #Discounted... https://t.co/nLCs0u0baM
1.06E+18	11/17/2018	7:43:04	FALSE	0	This Russian man pays for #iPhone XS with coins. <U+0001F92A> https://t.co/NdpeJQDKwt
1.06E+18	11/17/2018	7:42:37	FALSE	0	Surround sound explained https://t.co/NFwdqfnXH3 #tech #innovation #technology #iphonexs #news
1.06E+18	11/17/2018	7:36:42	FALSE	0	#Bible verse found with Words of Jesus Each Day by @RobotiCode for #Android #iPhone #Kindle. #DailyInspiration https://t.co/45v2ug0XKo
1.06E+18	11/17/2018	7:35:49	FALSE	0	https://t.co/V4moeIFMA9 Do u still listen to music on iPod? Here is smarter iPod. #iphone #iphoneapp Demo Video ->... https://t.co/IxOcxSlqut
1.06E+18	11/17/2018	7:34:03	FALSE	0	you will like this! https://t.co/ITlBr1VbuR #PR #senden #mobile #iphone
1.06E+18	11/17/2018	7:30:09	FALSE	0	#Stuffcool Finesse Sync & Charge #Lightning Cable 1.2M (#Apple MFi Certified) 1 Year #Warranty ! #Gojojo
1.06E+18	11/17/2018	7:30:01	FALSE	6	Our best #iPhone deal!

Figure 7. Sample of tweets in structured format.

- Filtering – we eliminate useless parts of tweets, such as URL links, Twitter usernames, punctuations, hashtags, Twitter special words (such as "RT"), special characters and symbols.
- Stop words removal –some words, such as pronouns (he, she, it), articles (a, an, the), don't give any information for classification. Moreover, having these bags of words can lead to less accurate prediction. So, it's better to eliminate these stop words [43].
- Stemming – it is a process of conversion of words in different forms into their single root word like "amuse", "amused", "amusement" and "amusing" have same root: "amus". Result of stemming is less intuitive to humans, but more comparable across observations. Stemming decreases entropy and increases relevance of root words like "amus" [43].

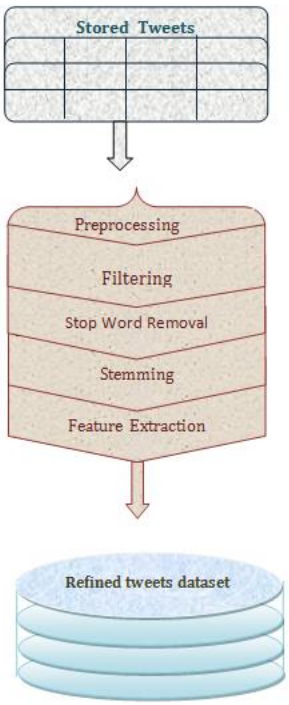


Figure 8. Pre-processing of Tweets.

- Feature extraction - Tokenization is a process of segmenting text by spaces and punctuation marks into tokens to form bags of words. Feature transformation function, like StringIndexer, OneHotEncoder and VectorIndexer, is used to transform categorical terms into vectors. TF-IDF is used to generate feature vectors from tweets. In TF-IDF, we compute TF (term

frequency), which is the occurrence frequency of a term in that document and IDF (inverse document frequency) measuring how infrequent a word is present across all the document. TF-IDF shows relevancy of a word into a specific document. Spark MLlib library has HashingTF and IDF algorithms to calculate TF-IDF [44]. Figure 8 shows the execution of pre-processing steps. After completion of data filtering steps, we get refined tweets with their labels. A sample of tweets with their polarity is shown in Figure 9.

#	Tweet	Class
1	upgradeupd acer liquid zmarshmallowliquid zhtml acer liquid...	Netural
2	kinder log into acer chromebook nd day school	Netural
3	readi yah acer desktop axc sffwinpro intel core™ i processorghzm cacheintel h...	Netural
4	acer chromebook– u intel celeron nghz gb ram gb flash chrome acer ...	Netural
5	so stress deallaptop i just may get shitfaced	Negative
6	as right now i like laptop im unsatisfi everi custom servic interact iv...	Positive
7	yep getfps fortnite	Netural
8	upgradeupd acer liquid jade zmarshmallowliquid jade zhtml acer li...	Netural
9	hot acer xfq bmiirxfull hd freesync game monitor overview	Positive
10	i love reinstal programs much fun do i get look forward everimonths	Positive

Figure 9. Sample of tweets with labels.

5.3 Tweet Score Calculation

This approach uses a standard list of positive and negative words to detect the polarity of a tweet. Based on availability of positive or negative words within tweets, a sentiment score is generated. Polarity of a tweet, such as p(t) can be represented as {-1,0,1} referring to a negative, neutral and positive tweet, respectively [45].

A score of a tweet S(t) can be calculated as:

$$S(t) = \sum_{i \in t}^{n} p(i) \tag{7}$$

where, p(i) is the polarity of term i in tweet. Polarity of a tweet can be determined as follows:

$$P(t) = \begin{cases} 1, \text{ if } St > 0 \text{ (positive)} \\ -1, \text{ if } St < 0 \text{ (negative)} \\ 0, \text{ otherwise (neutral)} \end{cases}$$

After score calculation for each tweet, we have training datasets with their polarities, such as positive, negative and neutral.

5.4 Model Implementation

ML is a dataframe package API, introduced in Spark 2.0. From start, spark framework has MLlib as an RDD-based API. To carry out the implementation in Spark, we need to follow some steps.

Firstly, import data into DataFrames. these are a distributed collection of data organized into named columns, which makes Spark programming easier and simpler to develop.

Secondly, transforms, such as Tokenizer (), StopWordRemover (), HashingTF (), Tf-Idf, are used. Transformer is an algorithm which can change one dataframe to another.

Thirdly, estimators are used to implement method fit(), which accept dataframe and make a model, such as logistic regression, Naïve Bayes, random forest, linear SVM and decision tree.

```
val Estimator = new LinearSVC()
val Estimator = new NaiveBayes().setLabelCol("label").setFeaturesCol("features")
val Estimator = new LogisticRegression()
```

Lastly, to combine ML algorithms into a single pipeline, we use Spark ML standardize APIs. Pipeline chains multiple transformers and estimators together in order to specify an ML workflow.

val pipeline = new Pipeline().setStages(Array(labelIndexer, tokenizer, remover, hashingTF, idf, Estimator))

val model = pipeline.fit(training)

In this classification step, to train the model, 70% of the dataset is randomly selected for training and 30% for testing.

val predictions = model.transform(test)

6. RESULTS AND DISCUSSION

This section describes the details of experiments conducted on the Spark framework.

6.1 Environment Description

We conducted experimental tests on Spark framework using a single node configuration. To achieve the desired performance, we have operated on Intel quad-core 3.0 GHz processor with a RAM of 8 GB and a storage capacity of 1 TB on Ubuntu 18.0.1 operating system. We configured Spark version 2.3.0, Scala version 2.11.6, Hadoop version 2.8.4, Flume 1.7.0, Hive 2.1.1 and Java-8.

We have used three different types of dataset related to electronic products; i.e., mobile phones, laptops and televisions, corresponding to 100 K, 70K and 50K tweets.

6.2 Polarity of Datasets

In this section, we have a pictorial representation of polarity in relation with phone, laptop and television tweets. Figures 10, 11 and 12 show the polarity of datasets indicating the ratio of positive, neutral and negative tweets, respectively.

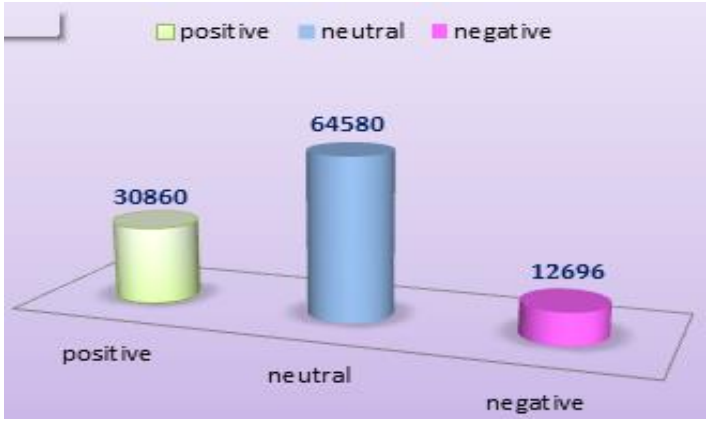


Figure 10. Polarity of phone dataset.

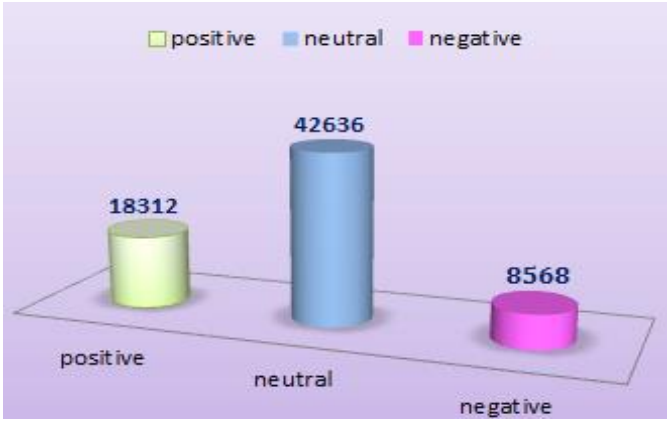


Figure 11. Polarity of Laptop Datasets.

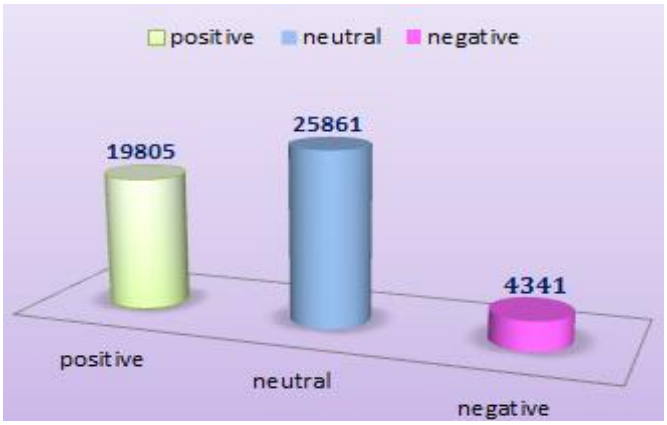


Figure 12. Polarity of television dataset.

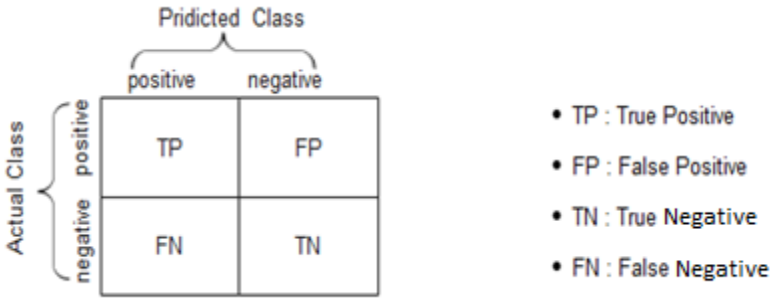
6.3 Performance Evaluation

Before the model can be used to classify new data, evaluation of model on test dataset is important. To measure the effectiveness or quality of models, different metrics are being used.

The simplest model of evaluation metric is precision. It measures the exactness of the model. It calculates what fraction of positive classified data is actually positive. Recall is another simple measurement. It measures the completeness of the model. It calculates what percentage of positive data is classified as positive. Accuracy measures what fraction of data is accurately classified. F-measure and AUC are commonly used metrics for model evaluation. F-measure is the weighted harmonic mean of precision and recall. It is the trade-off between precision and recall, whose score lies between 0 and 1. F-measure with score 1 states the best model whereas 0 states the worst model.

AUC (area under ROC): It is a binary classifier generally evaluated using AUC evaluation metric. It measures the aggregate performance with every classification parameter. It plots true positive rate and false positive rate at random positive or negative observations. Table 3 shows the confusion matrix, which is a specific table layout that allows visualization of the effectiveness of a model.

Table 3. Confusion matrix.



$$\text{Precision} = \frac{TP}{TP+FP} \tag{7}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{8}$$

$$\text{Accuracy} = \frac{TP+FN}{TP+FP+FP+TN} \tag{9}$$

$$F1 = \frac{2.\text{Precision}.\text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

Furthermore, the performance of different machine learning classification approaches is shown in Table 4, Table 5 and Table 6, respectively.

Table 4. Performance comparison (phone dataset).

Classification Approach	Accuracy	Recall	F1-measure	Precision
Naïve Bayes	0.82277	0.82277	0.82639	0.83386
SVM	0.85200	0.85200	0.84441	0.85011
Logistic Regression	**0.86358**	**0.86358**	**0.86006**	**0.86054**
Decision Tree	0.74882	0.74882	0.67685	0.79132
Random Forest	0.73647	0.73647	0.64792	0.79835

Table 5. Performance comparison (television dataset).

Classification Approach	Accuracy	Recall	F1-measure	Precision
Naïve Bayes	0.81973	0.81973	0.83528	0.87702
SVM	0.89777	0.89771	0.89098	0.89178
Logistic Regression	**0.91084**	**0.91084**	**0.90813**	**0.90724**
Decision Tree	0.81713	0.81713	0.73632	0.75132
Random Forest	0.81713	0.81713	0.73632	0.75132

Table 6. Performance comparison (laptop dataset).

Classification Approach	Accuracy	Recall	F1-measure	Precision
Naïve Bayes	0.81027	0.81027	0.81552	0.83448
SVM	0.86609	0.86609	0.86328	0.86434
Logistic Regression	**0.91084**	**0.91084**	**0.90813**	**0.90724**
Decision Tree	0.70493	0.70493	0.60479	0.76341
Random Forest	0.68892	0.68892	0.57204	0.77147

6.4 Comparison of Different Machine Learning Approaches

In this subsection, we have performed a series of tests using different machine learning classification approaches under the big data framework on our dataset. This comparison is carried out under different parameters. Figures 13 and 14 show the comparison of varied approaches in relation to training and prediction time on different datasets.

Figure 13 shows that for training the model, Naïve Bayes classifier takes less time related to all three categories. Similarly, to prepare the model, random forest classifier takes more time. It also informs that there is a direct relation between tweet size and training time; i.e., as tweet size increases, training time also increases.

Prediction time comparison using all approaches is shown in Figure 14. We can further conclude that logistic regression takes more prediction time in all three cases, while all the remaining approaches take approximately the same prediction time. Figure 15 shows accuracy comparison of all the approaches. This figure illustrates that logistic regression performs better for larger data sizes with an accuracy of 86% in the phone, 91% in the laptop and 91% in the television classes.

Another comparison measure is AUC (Area under the curve). The comparative result set value is shown in Table 7. It determines which approach best predicts the classes. Based on this view, Figure 16 shows that both SVM and logistic regression classification approaches perform good, compared to the other approaches.

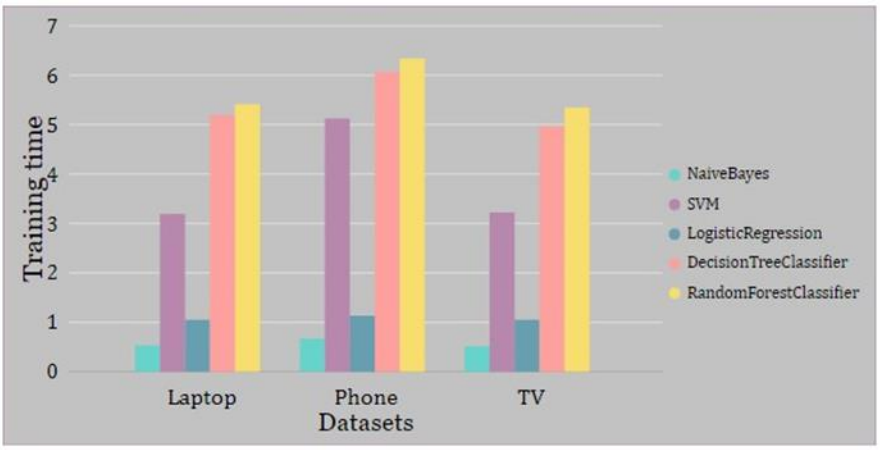


Figure 13. Training time comparison.

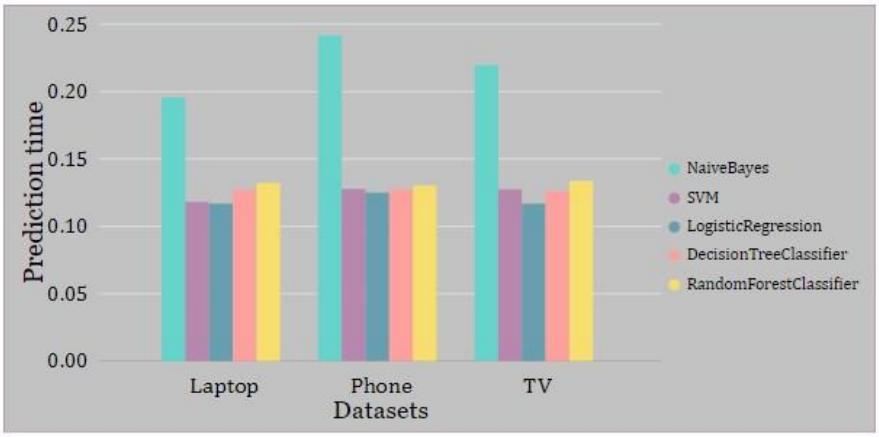


Figure 14. Prediction time comparison.

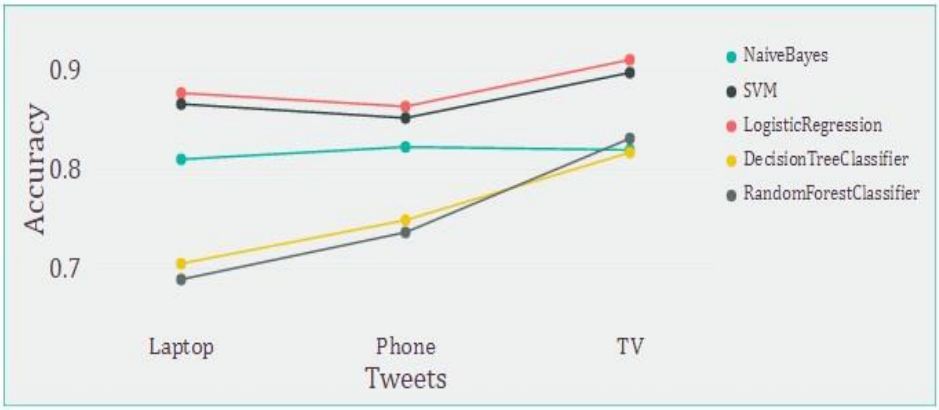


Figure 15. Accuracy comparison.

7. CONCLUSION AND FUTURE WORK

In this paper, we analyze sentiments of different electronic product tweets. For this, real-time tweets are collected from the Twitter platform using different hashtags. Additionally, Flume was used to consume real-time tweets in big data framework. After pre-processing of collected tweets, sentimental

Table 7. AUC results.

Classification Approach	Laptop	TV	Phone
Naïve Bayes	0.5028828	0.3904879	0.5012804
SVM	0.9277823	0.9281304	0.9100904
Logistic Regression	**0.9357323**	**0.9475399**	**0.9200218**
Decision Tree Classifier	0.6800121	0.7552956	0.5545057
Random Forest Classifier	0.8290897	0.8136763	0.8095101

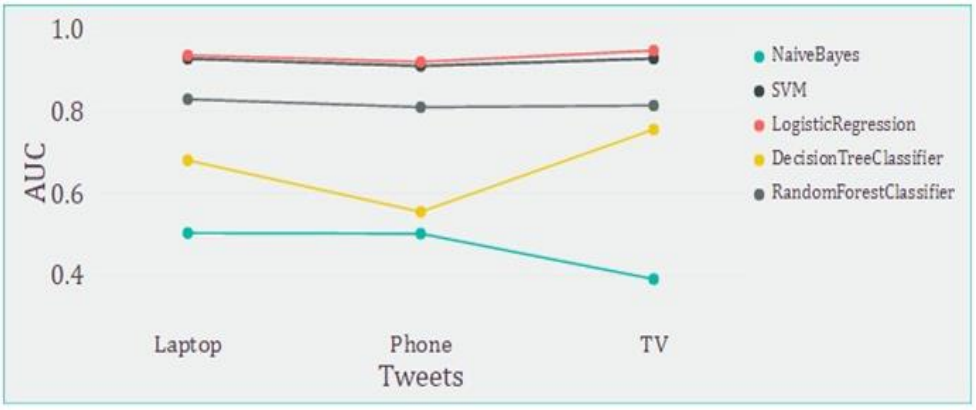


Figure 16. AUC comparison.

analysis has been performed by different supervised classification approaches. The experimental results show that the logistic regression approach has higher accuracy for all used datasets. Sentimental analysis comparison was carried out on the basis of Accuracy, F-measure and AUC.

Due to enhancement and popularity of social media platforms, such comparative results are more useful for business companies. They can easily help identify people's sentiment towards any specific electronic product or item. Based on sentiments, various decisions can be made.

In our future work, we intend to work on multiclass approaches to identify the exact polarity of tweets instead of positive, negative and neutral. In addition, we will work to enhance the accuracy of the approaches under big data technologies.

REFERENCES

[1] B. Liu, "Sentiment Analysis and Opinion Mining," Synthesis Lectures on Human Language Technologies, vol. 5, no. 1, pp. 1–167, 2012.

[2] A. Hasan, S. Moin, A. Karim and S. Shamshirband, "Machine Learning-based Sentiment Analysis for Twitter Accounts," Mathematical and Computational Applications, vol. 23, no. 1, p. 11, 2018.

[3] C. S. Khoo and S. B. Johnkhan, "Lexicon-based Sentiment Analysis: Comparative Evaluation of Six Sentiment Lexicons," Journal of Information Science, vol. 44, no. 4, pp. 491–511, 2017.

[4] F. Iqbal, J. Maqbool, B. C. M. Fung, R. Batool, A. M. Khattak, S. Aleem and P. C. K. Hung, "A Hybrid Framework for Sentiment Analysis Using Genetic Algorithm-based Feature Reduction," IEEE Access, pp. 1–1, 2019.

[5] F. Atefeh and D. Inkpen, Proceedings of the Workshop on Semantic Analysis in Social Media, Association for Computational Linguistics, France, 2012.

[6] A. Tyagi and S. Naresh, "Sentiments Analysis of Twitter Data Using K-Nearest Neighbour Classifier," International Journal of Engineering Science, vol. 17258, 2018.

[7] T. White, Hadoop: The Definitive Guide, 3rd Edition, O'Reilly Media, Inc., May 2012.

[8] M. V. Banerveld, N.-A. Le-Khac and M.-T. Kechadi, "Performance Evaluation of a Natural Language Processing Approach Applied in White Collar Crime Investigation," Future Data and Security Engineering Lecture Notes in Computer Science, pp. 29–43, 2014.

[9] G. Dubey, S. Chawla and K. Kaur, "Social Media Opinion Analysis for Indian Political Diplomats," Proc. of the IEEE 7th International Conference on Cloud Computing, Data Science and Engineering-Confluence, pp. 681-686, 2017.

[10] S. Al-Saqqa, G. Al-Naymat and A. Awajan, "A Large-Scale Sentiment Data Classification for Online Reviews Under Apache Spark," Procedia Computer Science, vol. 141, pp. 183–189, 2018.

[11] R. Sandy, U. Laserson, S. Owen and J. Wills, Advanced Analytics with Spark: Patterns for Learning from Data at Scale, O'Reilly Media, Inc., 2017.

[12] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross and K. J. Miller, "Introduction to WordNet: An On-line Lexical Database," International Journal of Lexicography, vol. 3, no. 4, pp. 235–244, 1990.

[13] A. Esuli and F. Sebastiani, "SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining," [Online], Available: http://nmis.isti.cnr.it/sebastiani/Publications/LREC06.pdf, 2006.

[14] S. Seifollahi and M. Shajari, "Word Sense Disambiguation Application in Sentiment Analysis of News Headlines: An Applied Approach to FOREX Market Prediction," Journal of Intelligent Information Systems, vol. 52, no. 1, pp. 57–83, 2018.

[15] M. Bhuiyan, A. Misra, S. Tripathy, J. Mahmud and R. Akkiraju, "Don't Get Lost in Negation: An Effective Negation Handled Dialogue Acts Prediction Algorithm for Twitter Customer Service Conversations," arXiv preprint arXiv:1807.06107, 2018.

[16] S. -M. Kim and E. Hovy, "Determining the Sentiment of Opinions," Proceedings of the 20th International Conference on Computational Linguistics (COLING 04), [Online], Available: http://aclweb.org/anthology/C04-1200, 2004.

[17] T. Wilson, J. Wiebe and P. Hoffmann, "Recognizing Contextual Polarity in Phrase-level Sentiment Analysis," Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT 05), [Online], Available: https://people.cs.pitt.edu/~wiebe/pubs/papers/emnlp05polarity.pdf, 2005.

[18] J. Blitzez, M. Dredze and F. Pereira, "Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification," Proc. of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 440–447, 2007.

[21] J. Kim, M. Yang, Y. Hwang, S. Jeon, K. Kim, I. Jung, C. Choi, W. Cho and J. Na, "Customer Preference Analysis Based on SNS Data," Proc. of the IEEE 2nd International Conference on Cloud and Green Computing, pp. 609-613, 2012.

[22] M. Kumar and A. Bala, "Analyzing Twitter Sentiments through Big Data," Proc. of the IEEE 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, pp. 2628-2631, 2016.

[24] A. L. Berger, V. J. D. Pietra and S. A. D. Pietra, "A Maximum Entropy Approach to Natural Language Processing," Computational Linguist, vol. 22, no. 1, pp. 39–71, 1996.

[25] A. Agarwal, B. Xie, I. Vovsha, O. Rambow and R. Passonneau. "Sentiment Analysis of Twitter Data," Proceedings of the Workshop on Languages in Social Media, pp. 30-38, 2011.

[26] F. H. Khan, S. Bashir and U. Qamar, "TOM: Twitter Opinion Mining Framework Using Hybrid Classification Scheme," Decision Support Systems, vol. 57, pp. 245–257, 2014.

[27] S. Geetha and K. V. Kumar, "Tweet Analysis Based on Distinct Opinion of Social Media Users," Advances in Intelligent Systems and Computing Advances in Big Data and Cloud Computing, pp. 251–261, 2018.

[28] A. Kaur, D. Khaneja, K. Vyas and R. S. Saini, Sentiment Analysis on Twitter Using Apache Spark, [Online], Available: https://www.researchgate.net/profile/Deepesh_Khaneja/publication/320625064_project_report_sentiment_analysis_on_twitter_using_apache_spark/links/59f24420aca272cdc7d0169a/project-report-sentiment-analysis-on-twitter-using-apache-spark.pdf, 2016.

[29] R. Kaptein, "Learning to Analyze Relevancy and Polarity of Tweets," CLEF (Online Working Notes/Labs/Workshop), [Online], Available: http://ceur-ws.org/Vol-1178/CLEF2012wn-RepLab-Kaptein2012.pdf, 2012.

[30] A. Kanavos, N. Nodarakis, S. Sioutas, A. Tsakalidis, D. Tsolis and G. Tzimas, "Large Scale Implementations for Twitter Sentiment Classification," Algorithms, vol. 10, no. 1, p. 33, 2017.

[31] A. Baltas, A. Kanavos and A. K. Tsakalidis, "An Apache Spark Implementation for Sentiment Analysis

on Twitter Data," Algorithmic Aspects of Cloud Computing Lecture Notes in Computer Science, pp. 15–25, 2017.

[32] W. N. Chan and T. Thein, "A Comparative Study of Machine Learning Techniques for Real-time Multi-tier Sentiment Analysis," Proc. of the IEEE 1st International Conference on Knowledge, Innovation and Invention (ICKII), 2018.

[33] N. Deshai, S. Venkataramana and G. P. S. Varma, "Performance and Cost Evolution of Dynamic Increase Hadoop Workloads of Various Data Centers," Smart Intelligent Computing and Applications Smart Innovation, Systems and Technologies, pp. 505–516, 2018.

[34] J. Dean and S. Ghemawat, "MapReduce," Communications of the ACM, vol. 51, no. 1, p. 107, 2008.

[35] M. Zaharia et al., "Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing, " Proc. of the 9th USENIX Conference on Networked Systems Design and Implementation, pp. 2-2, 2012.

[36] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu and D. Xin, "MLlib: Machine Learning in Apache Spark," The Journal of Machine Learning Research, vol. 17, no. 1, pp. 1235-1241, 2016.

[37] J. Hellerstein, J. Thathachar and I. Rish, "Recognizing End-user Transactions in Performance Management," Proc. AAAI-2000, pp. 596–602, 2000.

[38] J. Han, M. Kamber and J. Pei, "Data Mining: Concepts and Techniques," Elsevier, pp. 279–325, 2012.

[39] V. Vapnik, Estimation of Dependencies Based on Empirical Data, ISBN 978-0-387-34239-9, Springer, 1995.

[40] K. Karimi and J. H. Howard, "Generation and Interpretation of Temporal Decision Rules," arXiv preprint arXiv:1004.3334, 2010.

[41] L. Breiman, "Random Forests," UC Berkeley TR567, 1999.

[43] G. Angiani, L. Ferrari, T. Fontanini, P. Fornacciari, E. Iotti, F. Magliani and S. Manicardi, "A Comparison between Pre-processing Techniques for Sentiment Analysis in Twitter," In: KDWeb, 2016.

[44] H. Karau, A. Konwinski, P. Wendell and M. Zaharia, Learning Spark: Lightning-fast Big Data Analysis, O'Reilly Media, Inc., Jan 2015.

[45] A. Giachanou, J. Gonzalo, I. Mele and F. Crestani, "Sentiment Propagation for Predicting Reputation Polarity," Lecture Notes in Computer Science Advances in Information Retrieval, pp. 226–238, 2017.

ملخص البحث:

في وقتنا الحاضر، أصبحت وسائل التواصل الاجتماعي شائعةً جداً في ظل التقدم الذي طرأ على تقنيات الانترنت وأجهزة الهاتف الذكية. وقد أحدثت منصات التواصل الاجتماعي اهتماماً لافتاً بين المستخدمين لإبداء آرائهم. كذلك تلعب وسائل التواصل الاجتماعي، مثل تويتر، دوراً مهماً لشركات الأعمال. وبناءً على آراء الزبائن حول منتج ما، تصبح الشركات على اطلاع على اختيارات الزبائن. وفي الوقت الراهن، تتولد ملايين التغريدات من قبل الناس كل سنة. إلا أن التعامل مع هذا الكم الهائل من التغريدات غير المهيكلة ليس ممكناً عبر المنصة التقليدية. لذا، فإن أحد الأطر الخاصة بمعالجة البيانات الضخمة – مثل هادوب وسبارك – يستخدم من أجل التعامل مع هذا النوع من البيانات الضخمة.

في هذه الورقة، يتم استخدام تغريدات تتعلق بالمبيعات لتحليل آراء الزبائن بشأن المنتجات الالكترونية. وتجدر الإشارة الى أن النتائج التجريبية للعمل المقترح ستكون ذات فائدة للعديد من شركات الأعمال بحيث تساعدها في اتخاذ القرارات المتعلقة بأعمالها، الأمر الذي من شأنه أن يعمل على تحسين مبيعاتها من المنتجات.

المجلة الأردنية للحاسوب و تكنولوجيا المعلومات

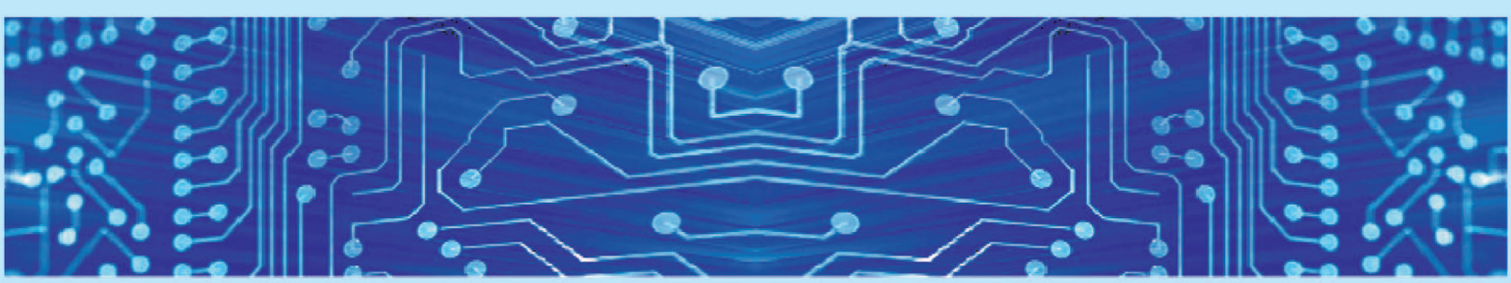
ISSN 2415 - 1076 (Online)
ISSN 2413 - 9351 (Print)

نيسان ٢٠١٩ المجلد ٥ العدد ١

JJCIT

www.jjcit.org

jjcit@psut.edu.jo



مجلة علمية عالمية متخصصة محكمة

تصدر بدعم من صندوق دعم البحث العلمي