Quantum-Inspired Evolutionary Algorithm to Solve Graph Coloring Problem

Pronaya Prosun Das, Mozammel H. A. Khan

Abstract—Graph Coloring Problem (GCP) bears an enormous significance to the researchers in the field of soft computing. In this paper, we demonstrate a Quantum-Inspired Evolutionary Algorithm (QEA) to solve GCP. We use two dimensional arrays of Q-bits called Q-bit individual to produce binary individual. Q-gate operation is applied as a variation operator on Q-bit individuals. In traditional evolutionary algorithm (EA) for GCP, k-coloring approach is used and the EA is run several times for decreasing value of k until lowest possible k is reached. In our QEA, we start with the number of colors equal to the theoretical upper bound of the chromatic number, which is maximum out degree + 1, and during evolution some colors are made unused to reduce the number of color in each generation. As a result, solution is found in a single run. We test 36 datasets from DIMACS benchmark and compare the result with several recent works. For five datasets, our algorithm obtains better solution than other.

Keywords—quantum-inspired evolutionary algorithm (QEA), graph coloring problem (GCP), combinatorial optimization, Q-bit representation, Q-gate.

I. Introduction

Graph coloring problem (GCP) is a well-known NP-hard problem [1]. GCP assigns different colors to the adjacent vertices of a graph using minimum number of colors. The GCP is illustrated with a simple graph in Fig. 1, where minimum number of colors needed to color eleven vertices is three. In Fig. 1, the vertex number is shown within the circle and the color number is shown outside the circle. The notable application of GCP are seen in pattern recognition [2], map coloring [3], radio frequency assignment [4], Bandwidth allocation [5], and timetable scheduling [6] etc.

![Example of graph coloring.](image)

Our objective is to color all the vertices initially with \( m = d + 1 \) and then reducing \( m \) dynamically so that minimum chromatic number, denoted by \( x(G) \), is found, that is \( m = x(G) \) is reached. In this paper, we propose a Quantum-inspired Evolutionary Algorithm (QEA) [7], which is the combination of concept of quantum computing [8] and evolutionary algorithm. It uses population of Q-bit individuals and Q-gate as a main variation operator.

II. Prior Work

One of the most recent works on GCP is Memetic Algorithm (MA) [11] that used binary encoded chromosome. Population is updated mainly using classical crossover operator. Offsprings are corrected if needed. Then a deterministic improvement technique is applied on the corrected offsprings to improve the solution quality. Another work on GCP [12] combines wisdom of artificial crowds approach with the genetic algorithm (GA). In this approach, multiple parent selection and multiple mutations based on the closeness of the solution to the global optima are used. The algorithm is run several times for several decreasing values of \( k \) and the minimum possible \( k \) value is taken as the minimum chromatic number. A guided genetic algorithm for GCP called MSPGCA is reported in [13], where the authors fine-tuned the initial chromosmes using a simple genetic algorithm and then the deterministic MSPGCA algorithm is run to dynamically reduce the chromatic number. In paper [14], the authors have proposed a hybrid algorithm to solve GCP. GA has a highly degenerate objective function. In order to compensate for this degeneracy, bitstream neuron (Boltzmann Machine) was applied to the solution obtained from GA. A hybrid immune algorithm is also applied on GCP [15]. All the above mentioned approaches used integer encoding for the chromosomes except the paper [11].

III. Methodology

Representation of the Graph is discussed first and then the proposed QEA for GCP is discussed.

A. Representation

A Q-bit is defined as the smallest unit of information [8] in QEA, which is defined with a pair of numbers \((\alpha, \beta)\), where \(|\alpha|^2 + |\beta|^2 = 1\). \(|\alpha|^2\) and \(|\beta|^2\) gives the probability that the Q-bit will be found in the “0” state and the “1” state, respectively. For GCP, we use two-dimensional array of Q-bits as a Q-bit individual, where each row corresponds to a color and each column corresponds to a vertex. Later binary individuals are produced from Q-bit individuals. If the \(j\)th vertex be colored using the \(i\)th color, then the \((i, j)\)th element of the array is 1 and the other elements are 0s. Thus, in a valid chromosome, every column must have a single 1 and a row
will have one or more than one 1s placed on non-adjacent vertices columns. The encoding scheme is illustrated in Fig. 2. A row may also have all 0s, in which case the color is not used in the solution. If a column has all 0s (the vertex is not colored) or more than one 1s (more than one color is assigned to that vertex), then the encoding is invalid. On the other hand, if a row has 1s in adjacent vertices columns (same color is assigned to adjacent vertices), then the encoding is invalid. These situations may arise during creation of the binary individual from Q-bit individual. When a binary individual becomes invalid, then it is corrected using repair procedure. Thus the number of rows having at least one 1 is the number of used colors and is used as the fitness function in our QEA.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 2: Proposed binary individual for GCP of the graph of Fig. 1.

B. Proposed QEA to solve GCP

We present the detailed algorithm of QEA for the Graph Coloring. The Graph Coloring problem is considered to demonstrate the applicability of QEA to the combinatorial optimization problem.

Procedure QEA for the GCP

01: begin
02: t ← 0
03: initialize Q(t)
04: while (t < MAX GEN) do
05: begin
06: i ← 0
07: while (i < m) do
08: begin
09: j ← 0
10: while (j < v) do
11: if (random [0, 1) < |β_i,j|^2)
12: then x_{i,j} ← 1
13: else x_{i,j} ← 0
14: end
15: end
16: end
17: t ← t + 1
18: make P(t) by observing the states of Q(t - 1)
19: repair P(t)
20: evaluate P(t)
21: update Q(t)
22: store the best solutions among P(t) into B(t)
23: end
24: while (t < MAX GEN) do
25: begin
26: i ← i + 1
27: j ← j + 1
28: end

Binary individuals are repaired if needed. Two possible problems may occur – (i) a column may have multiple 1s or (ii) a column may have all 0s. We also have to ensure that adjacent vertices have colored with different colors.

The chromosome shown in Fig. 3 is invalid and has the two possible problems. Invalid chromosomes are corrected by repair procedure. For case (i), only one 1 is kept and for case

q_p^t = \begin{bmatrix} a_p^t_{1,1} & a_p^t_{1,2} & … & a_p^t_{1,v} \\ a_p^t_{2,1} & a_p^t_{2,2} & … & a_p^t_{2,v} \\ . & . & . & . \\ a_p^t_{m,1} & a_p^t_{m,2} & … & a_p^t_{m,v} \end{bmatrix}

Where, m × v is the number of Q-bits in an individual and p = 1, 2, ..., k. In the step of initialize Q(t), all α_i,j and β_i,j are initialized with 1/√2, where, i = 1, 2, ..., v and j = 1, 2, ..., m. On line 04 and 11, QEA produce population of binary individuals, P(t) = \{x_1^t, x_2^t, ..., x_v^t\} from population of Q-bit individuals, where t = 0, 1, 2, ... . For notational simplicity, x and q are used instead of x_i^t and q_p^t respectively. The following make procedure is used to obtain the binary string x.
(ii), a 1 is inserted. Both are done at a selected row among used color cluster which is sorted according to number of uses so that no conflict is created at that row (breaking tie randomly). If such a used color row is not available, then a 1 is inserted at a randomly selected row among unused color clusters. A possible corrected version of invalid chromosome of Fig. 3 is shown in Fig. 4.

<p>| | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 4: Corrected Chromosome of Fig. 3.

Binary chromosomes are corrected using repair procedure as follows:

**Procedure Repair (x)**

1: begin
2: for (all vertex v in random order) do
3: begin
4: iscolored ← false
5: for (all color i which is assigned to v in a ordering most to least used) do
6: begin
7: if (iscolored) then remove color i from v
8: else if (v is colorable with i) then iscolored ← true
9: end
10: end
11: end
12: end
13: for (all vertex v which is not colored) do
14: begin
15: iscolored ← false
16: for (all used color i in a ordering most to least used) do
17: begin
18: if (not iscolored and v is colorable with i) then
19: color the vertex, v with i
20: iscolored ← true
21: end
22: end
23: end
24: end
25: end
26: then color the vertex v with an unused color
27: end
28: end

The update procedure of Q-bits is presented below:

**Procedure Update (q)**

1: begin
2: i ← 0
3: j ← 0
4: while (i < m) do
5: begin
6: i ← i + 1
7: while (j < v) do
8: begin
9: j ← j + 1
10: Determine Δθ_{i,j} with the lookup table
11: Obtain (α_{i,j}, β_{i,j}) from the following:
12: if (q is located in the first/third quadrant) then
13: [α_{i,j}', β_{i,j}'] = H_e(α_{i,j}, β_{i,j}, Δθ_{i,j})
14: else [α_{i,j}', β_{i,j}'] = H_e(α_{i,j}, β_{i,j}, -Δθ_{i,j})
15: end
16: end
17: q ← q'
18: end

Here, H_e gate is used as a Q-gate to update a Q-bit individual q as a variation operator. Q-bit of i-th row and j-th column (α_{i,j}, β_{i,j}) is updated as follows:

\[
[α_{i,j}', β_{i,j}'] = U(Δθ_{i,j})[α_{i,j}, β_{i,j}]
\]

Where, \(U(Δθ_{i,j})\) is a simple rotation gate,

\[
U(Δθ_{i,j}) = \begin{bmatrix}
\cos(Δθ_{i,j}) & -\sin(Δθ_{i,j}) \\
\sin(Δθ_{i,j}) & \cos(Δθ_{i,j})
\end{bmatrix}
\]

i) if \(|α_{i,j}'|^2 < 1 - ε\) and \(|β_{i,j}'|^2 < 1 - ε\) then
\[
[α_{i,j}', β_{i,j}'] = [\sqrt{1-ε}, 1 - \sqrt{1-ε}]
\]

ii) if \(|α_{i,j}'|^2 < 1 - ε\) and \(|β_{i,j}'|^2 < ε\) then
\[
[α_{i,j}', β_{i,j}'] = [\sqrt{1-ε}, \sqrt{ε}]
\]

iii) otherwise
\[
[α_{i,j}', β_{i,j}'] = [α_{i,j}, β_{i,j}]
\]

Fig. 5: \(H_e\) gate based on rotation gate.

In this QEA for GCP, the angle parameters used for the rotation gate are shown in Table 1. Let us define an angle vector \(θ = [θ_1, θ_2, ..., θ_8]^T\), where \(θ_1, θ_2, ..., θ_8\) can be found from Table 1. We have used, \(θ_3 = 0.01π, θ_5 = -0.01π,\) and 0 for the rest. The values from 0.001π to 0.05π are recommended for the magnitude of Δθ_{i,j}. Otherwise, it may converse prematurely. The sign of Δθ_{i,j} determines the
direction of convergence. We have chosen \( \epsilon = 0.01 \). In Table 1, \( f(\cdot) \) is the fitness, and \( x_{ij} \) and \( b_{ij} \) are the \((i, j)\)th bits of the best solution \( x \) and the binary solution \( b \), respectively. In the QEA for GCP, \( \theta_1 = 0, \theta_2 = 0, \theta_3 = 0.01\pi, \theta_4 = 0, \theta_5 = -0.01\pi, \theta_6 = 0, \theta_7 = 0, \theta_8 = 0 \) are used.

Table 1: lookup table of \( \Delta \theta_{ij} \)

<table>
<thead>
<tr>
<th>( x_{ij} )</th>
<th>( b_{ij} )</th>
<th>( f(x) \geq f(b) )</th>
<th>( \Delta \theta_{ij} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>false</td>
<td>( \theta_1 )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>true</td>
<td>( \theta_2 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>false</td>
<td>( \theta_3 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>( \theta_4 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>false</td>
<td>( \theta_5 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>true</td>
<td>( \theta_6 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>false</td>
<td>( \theta_7 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>true</td>
<td>( \theta_8 )</td>
</tr>
</tbody>
</table>

In line 18 of Procedure QEA for the GCP, migration is defined as the process of copying current best solution in binary population in place of previous solutions. A local migration is implemented by replacing some of the population by the best individual, while global migration is implemented by replacing all the solution by the best individual.

### iv. Results

We have implemented our algorithm in C++ programming language and compiled using 32 bit TDM-GCC compiler, version 4.8.1. Tests were run on a PC having following configuration:

**CPU:** Intel Core i3-2350M 2.30 GHz

**Memory:** 4 GB DDR3 1333MHz

**Operating System:** Windows 7 64-bit

Datasets used to test our QEA for GCP are taken from Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) benchmarking graph collection [9] and [10]. Instances ending in .col are in DIMACS standard format. Instances in .col.b are in compressed format. We have used datasets ending with .col extension. The top of the dataset heading resembling “p edge 11 20” means that graph has 11 vertices and 20 edges, where \( p \) denotes vertices. After that number of lines like “e 1 2” represent connection between two edges. We experiment with 36 datasets from [9] and [10]. The tested datasets are heterogeneous consisting of big graph like 5-FullIns_4.col having 1085 vertices, highly dense graph like miles1500.col, highly complex graph like queen10_10.col, and even simple graphs. Our results are tabulated in Table 2 and compared with other results. For 29 datasets, we found expected chromatic number as stated in [9] and [10]. For dataset queen10_10.col, 1-FullIns_3.col, 1-FullIns_5.col, 2-FullIns_5.col, 3-FullIns_3.col, 3-FullIns_4.col, 5-FullIns_4.col, no expected chromatic number is stated there. We get better result for 2-FullIns_5.col, 5-FullIns_4.col compared to the result of paper [14]. For queen8_8.col our result is 9, which is better than the result of papers [13] and [14]. Our QEA produced more optimal result for queen8_12.col than result of [13]. For queen10_10.col, we get chromatic number 12, whereas this number found in paper [11] and paper [13] are 13 and 14, respectively.

Table 2: Comparison of Obtained Results with Other Works

| Dataset     | \( |V| \) | \( |E| \) | \( x(G) \) | [12] | [13] | [14] | [11] | QEA |
|-------------|---------|---------|--------|------|------|------|------|------|
| mycie44.col | 23      | 71      | 5      | -    | 5    | 5    |      |      |
| mycie55.col | 47      | 236     | 6      | -    | 6    | 6    |      |      |
| mycie66.col | 95      | 755     | 5      | -    | 7    | 7    |      |      |
| mycie77.col | 191     | 2360    | 8      | -    | 8    | 8    |      |      |
| games120.col | 120    | 638     | 9      | 9    | 9    | 9    |      |      |
| huck.col   | 74      | 301     | 11     | 11   | 11   | 11   |      |      |
| jean.col   | 80      | 254     | 10     | 10   | 10   | 10   |      |      |
| david.col  | 87      | 406     | 11     | 11   | 11   | 11   |      |      |
| queen5_5.col | 25     | 160     | 5      | 5    | 5    | 5    |      |      |
| queen6_6.col | 36     | 290     | 7      | 7    | 8    | 7    |      |      |
| queen7_7.col | 49     | 476     | 7      | 7    | 7    | 7    |      |      |
| queen8_8.col | 64     | 728     | 9      | 11   | 11   | 11   |      |      |
| queen8_12.col | 96    | 1368    | 12     | 14   | -    | -    | 12   |      |
| queen10_10.col | 100   | 2940    | ?      | 14   | 13   | 13   |      |      |
| anna.col   | 138     | 493     | 11     | 11   | 11   | 11   |      |      |
| homer.col  | 561     | 1629    | 13     | 13   | 13   | 13   |      |      |
| miles250.col | 128    | 387     | 8      | 8    | 8    | 8    |      |      |
| miles500.col | 128    | 1170    | 20     | -    | 20   | 20   |      |      |
| miles750.col | 128    | 4226    | 31     | 31   | 31   | 31   |      |      |
| miles1000.col | 128   | 3216    | 42     | 42   | 42   | 42   |      |      |
| miles1500.col | 128   | 5198    | 73     | 73   | 73   | 73   |      |      |
| zeroin.1.i.col | 211   | 4100    | 49     | -    | -    | -    | 49   | 49   |
| zeroin.2.i.col | 211   | 3541    | 30     | -    | -    | -    | 30   |      |
| zeroin.3.i.col | 206   | 3540    | 30     | -    | -    | -    | 30   |      |
| 2-Insetion_3.col | 37    | 72      | 4      | -    | 4    | 4    |      |      |
| initfix.1.i.col | 864   | 18707   | 54     | -    | -    | -    | 54   |      |
| initfix.2.i.col | 645   | 13979   | 31     | -    | -    | -    | 31   |      |
| mulsol.1.i.col | 197   | 3925    | 49     | -    | 49   | 49   |      |      |
| fpsol2.1.i.col | 496   | 11654   | 65     | 65   | -    | -    | 65   |      |
| mulsol.5.i.col | 186   | 3973    | 31     | -    | -    | -    | 31   |      |
| 1-FullIns_3.col | 30    | 100     | ?      | -    | -    | -    | 4    |      |
| 1-FullIns_5.col | 282   | 3247    | ?      | 6    | 6    | 6    |      |      |
| 2-FullIns_5.col | 852   | 12201   | ?      | -    | 11   | 11   | 7    |      |
| 3-FullIns_3.col | 80    | 346     | ?      | -    | -    | -    | 6    |      |
| 3-FullIns_4.col | 405   | 3524    | ?      | 7    | 7    | 7    |      |      |
| 5-FullIns_4.col | 1085  | 11395   | ?      | -    | 11   | 11   | 10   |      |

Fig. 6 shows the average fitness (number of used color) and minimum fitness over successive generation for queen7_7 dataset indicating the dynamicity of our algorithm. In our experiments, we found that rotation probability of 0.7 performs better for all datasets. The termination condition also depends on the graph complexity. If both the average fitness and the best fitness do not change for a specified number of generations or the optimal known solution is found, then the algorithm is terminated. The number of generations varies with the graph complexity. Our algorithm is very fast because it can work with small population and also it needs less
generation than other evolutionary algorithms to get the optimal solution. In our experiment, we have used population size within 5 to 50 for different datasets.

Fig. 6: Average and best (minimum) fitness for queen7_7 dataset.

v. Conclusion and Future Work

In this paper, we proposed a Quantum-inspired Evolutionary Algorithm (QEA) for graph coloring problem (GCP). The main variation operator of our QEA is the rotation gate. The population of the solution is updated using only rotation gate. We compare best binary individual with all binary individuals in the population and update the Q-bit individual. Because of the nature of the encoding, in each generation, binary individual may become invalid and in that case binary individuals are corrected to obtain the valid solution. After certain generations, all or partial population are replaced with best binary individual to avoid local optimization. We start with \( m = d + 1 \) colors, where \( d \) is the maximum out degree of the graph. The number \( m \) is the upper bound of the chromatic number. That means, in a single run, the QEA dynamically reduces the chromatic number starting with upper bound to the possible minimum number. Unlike the previous techniques, our QEA finds the minimum chromatic number in a single run reducing the total time significantly. We experiment with 36 datasets from [9] and [10]. For 29 datasets, we found expected chromatic number as stated in [9] and [10]. For five datasets queen8_8.col, queen8_12.col, queen10_10.col, 2-FullIns_5.col and 5-FullIns_4.col, our QEA produce better result than the previous works. So we can say these are the major achievement of our algorithm over the previous works. In the future, we will try to improve the execution time of the algorithm. We also have plans to compare more results of our proposed approach with the results produced by other algorithm.

References


About Author(s):

Pronaya Prosun Das was born in Manikganj, Bangladesh. He is a final year student of Computer Science and Engineering at East West University, Aftabnagar, Dhaka 1212, Bangladesh. His research interests include Evolutionary Algorithms, Machine Learning and Image Processing.

Mozammel H A Khan was born in Kushtia, Bangladesh. He obtained B. Sc. Engg. (EEE), M. Sc. Engg. (Comp. Engg.), and PhD (Comp. Sc. &Engg.) degrees from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh. He served as faculty member of EEE Department at Bangladesh Institute of Technology, Rajshahi, Bangladesh and Computer Science and Engineering Department at Khulna University, Khulna, Bangladesh. Currently, he is a full Professor of Computer Science and Engineering Department at East West University, Dhaka, Bangladesh. His research interest includes broad areas of Logic Synthesis, Reversible Logic, Multi-value Logic, and Soft Computing. He has published over 85 research papers. He is a Senior Member of IEEE.