CROSSOVER – A REDUNDANT OPERATOR FOR SOLVING SYSTEM OF LINEAR EQUATIONS BY UNIFORM ADAPTIVE HYBRID EVOLUTIONARY ALGORITHMS

A. R. M. Jalal Uddin Jamali¹, M. Arif Hossain¹ and M. M. A. Hashem²

¹Department of Mathematics, Khulna University of Engineering & Technology, Khulna-9203, Bangladesh
²Department of Computer Science and Engineering, Khulna University of Engineering & Technology, Bangladesh

Received: 10 October 2011  Accepted: 15 December 2011

ABSTRACT

In this paper we have studied the effectiveness and the necessity of the crossover operator for solving set of linear equations by Uniform Adaptive Hybrid Evolutionary Algorithm (HEA). Several experiments have been carried out for solving large set of linear equations by considering the Jacobi based as well as Gauss-Seidel based Uniform Adaptive hybrid algorithms and also their classical counterparts. We, here, mainly investigated the effectiveness of crossover by considering presence of crossover operation as well as absence of crossover operator in HEA. We also investigated the effectiveness of uniform adaptive HEA by comparing with classical ones. From this primary investigation it is found that the presence or absence of the crossover operation has no significant effect on uniform adaptive HEA. Moreover for the absence of crossover operation the proposed modified hybrid algorithms require less computational effort, and also they are more efficient, effective and easy to implement.

Keywords: Crossover, Evolutionary Algorithm, Mutation, Successive relaxation, and Uniform adaptation.

1. INTRODUCTION

Large set of linear equations frequently arise directly or indirectly in the real world problems such as in engineering and science, while we apply mathematics to the social sciences and for the quantitative study of business and commerce, statistical and economic problems. Though there are many classical methods available, scientists till have their keen interest to find out the methods, which converge rapidly and efficiently, for solving large set of linear equations. Furthermore efficient and rapid convergent methods are sometime desired for solving physical problems to make appropriate decision quickly. For example, solution of very large set of linear equations are required in the cases of short-term weather forecasting, image processing, simulation to predict aerodynamic performance and in all these cases time is an important factor for practical application of the results.

For solving large set of linear equations, especially for sparse and structured coefficients, iterative methods are preferable, as iterative method are unaffected by round off errors (Gerald and Wheatley, 2006). The rate of convergence of the well-known classical numerical iterative methods namely the Jacobi method and the Gauss-Seidel method is very slow and can be accelerated by using successive relaxation (SR) technique (Engeln and Uhlig, 1996). But the speed of convergence depends on the relaxation factor \( \omega \ (0 < \omega < 2) \) and SR technique is very much sensitive to the relaxation factor (Hagaman and Young, 1981). Moreover, most of the time, it is very difficult to estimate the optimal relaxation factor, which is a key parameter of the SR technique (Hagaman and Young, 1981; Gourdin and Boumahrat, 1996).

On the other hand the evolutionary algorithms (EA) are developed from some natural phenomena: genetic inheritance and Darwinian strife for survival (Back et al., 1997; Back and Schwefel, 1993; Schoenauer and Michalewicz, 1997). It is worthwhile to mention here a brief description of EA. The EA is a stochastic multisearch process which maintains a set of points (called population) that are searched in parallel. Each point (called individual) represents a potential solution to the problem and each individual is evaluated according to the objective function (called fitness function). There are two basic principles in evolution – selection and variation. The selection focuses the search to better regions of search space and variation creates new points in the search space. There exist two main mechanisms in variation namely (i) Mutation and (ii) Crossover. Mutation operator is applied to each individual of the population for random mild change of the search point. On the other hand crossover operator is applied to the all individuals of the population for random mixing of the information of individuals. To illustrate the crossover operator, consider two individuals (points) say \( x_1, x_2 \) and...
randomly generated a stochastic matrix $\mathbf{R}$, where $\mathbf{x}_1 = [a_1, a_2, a_3, a_4, a_5]$, $\mathbf{x}_2 = [b_1, b_2, b_3, b_4, b_5]$ and $\mathbf{R} = \begin{bmatrix} 0.4 & 0.6 \\ 0.7 & 0.3 \end{bmatrix}$. Then after crossover operation we have the offspring solution $\mathbf{x}_1'$ and $\mathbf{x}_2'$ as follows:

$$
\begin{bmatrix}
\mathbf{x}_1' \\
\mathbf{x}_2'
\end{bmatrix} =
\begin{bmatrix}
0.4 & 0.6 \\
0.7 & 0.3
\end{bmatrix}
\begin{bmatrix}
\mathbf{x}_1 \\
\mathbf{x}_2
\end{bmatrix} =
\begin{bmatrix}
0.4a_1 + 0.6b_1 & 0.4a_2 + 0.6b_2 & 0.4a_3 + 0.6b_3 & 0.4a_4 + 0.6b_4 & 0.4a_5 + 0.6b_5 \\
0.7a_1 + 0.3b_1 & 0.7a_2 + 0.3b_2 & 0.7a_3 + 0.3b_3 & 0.7a_4 + 0.3b_4 & 0.7a_5 + 0.3b_5
\end{bmatrix}
$$

i.e.

$$
\begin{bmatrix}
\mathbf{x}_1' \\
\mathbf{x}_2'
\end{bmatrix} =
\begin{bmatrix}
0.4a_1 + 0.6b_1 & 0.4a_2 + 0.6b_2 & 0.4a_3 + 0.6b_3 & 0.4a_4 + 0.6b_4 & 0.4a_5 + 0.6b_5 \\
0.7a_1 + 0.3b_1 & 0.7a_2 + 0.3b_2 & 0.7a_3 + 0.3b_3 & 0.7a_4 + 0.3b_4 & 0.7a_5 + 0.3b_5
\end{bmatrix}
$$

Note that any matrix whose each element is nonnegative and every row (or column) sum is one is called stochastic matrix. For more details it is referred to see Back and Schwefel (1993).

Generally, most of the works on EA can be classified as either evolutionary optimization (either numerical or combinatorial) or evolutionary learning (Hashem, 1999; Watanabe and Hashem, 2004; Michalewicz and Attia, 1996; Salomon and Hemmen, 1996). Recently, Gauss-Seidel based uniform adaptive (GSBUA) hybrid EA (Jun et al., 2000) and Jacobi based uniform adaptive (JBUA) hybrid EA (Jamali et al., 2003) have been developed for solving larger set of linear equations by integrating classical numerical methods with uniform adaptive (UA) evolutionary computation techniques. In these algorithms both crossover and mutations operations are used. Furthermore, Gauss-Seidel based Time variant adaptive (GSBTVA) hybrid evolutionary algorithm (Jamali et al., 2004a) and Jacobi based Time variant adaptive (JBTVA) hybrid evolutionary algorithm (Jamali et al., 2004b, Jamali et al., 2004c) have been developed for solving larger set of linear equations by integrating classical numerical methods with Time variant adaptive (TVA) evolutionary computation techniques. Both crossover as well as mutation operations are available in the later two algorithms. But recently it has been shown in (Jamali et al., 2005) that in time variant adaptive HEC (considering both Jacobi as well as Gauss-Seidel based approaches) crossover is a redundant step, i.e. only mutation operator (among the two) is enough, for solving linear equations.

It is also worthwhile to remark that the uniform adaptation or time variant adaptation techniques are introduced for self-adaptation of relaxation factor. The idea of self-adaptation was also applied in many different fields (Salomon and Van 1996, Jamali et al., 2009; Back 1997). It is also worthwhile to remark that uniform adaptation technique and time variant adaptation technique usually have significantly different impact in algorithm. Similarly, in general, Crossover and Mutation have also significantly different impact in evolutionary based algorithms.

In this paper, two modified uniform adaptive hybrid evolutionary algorithms (i.e. modified GSBUA and the modified JBUA algorithms) are proposed to solve larger set of linear equations in which crossover operator is absent. The proposed modified hybrid algorithms are obtained from the existing GSBUA and the JBUA algorithms (Jamali et al., 2004b), and contain all the evolutionary operations available in the existing algorithms except crossover step. The proposed modified hybrid algorithms initialize random relaxation factors in a given domain and “evolve” it by uniform adaptation technique as well. The main mechanisms of the proposed modified algorithms are initialization, mutation, uniform adaptation, and selection mechanisms (i.e. crossover operation is absent). It makes better use of a population by employing different equation-solving strategies for different individuals in the population. The errors are minimized by mutation and selection mechanisms. We have tried to investigate the effectiveness and necessity of the evolutionary operators mainly crossover operation available in the existing uniform adaptive hybrid evolutionary algorithms (Jun et al., 2000, Jamali et al., 2003) by comparing the proposed modified hybrid algorithms (which do not contain crossover operation) with existing hybrid algorithms (which contain both crossover). The primary investigation has showed that the both proposed modified algorithms are comparable with the corresponding existing hybrid evolutionary algorithms in terms of number of generation required to get expected result. Also the proposed modified algorithms are more efficient and effective than the corresponding existing GSBUA and JBUA algorithms in terms of computational effort (as they do not contain crossover operator). We also compared the proposed algorithms with the classical ones for showing the necessity of the HEA. It is worthwhile to remark that the proposed modified hybrid algorithms are relatively easier to implement and required less memory allocation.

2. THE BASIC EQUATIONS OF CLASSICAL METHODS

The system of $n$ linear equations with $n$ unknown $x_1, x_2, \cdots, x_n$ can be written as

$$
\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad (i = 1, 2, \cdots, n)
$$
or equivalently, in matrix form

\[ Ax = b \]  

where \( A \in \mathbb{R}^{n \times n}, x \in \mathbb{R}^{n} \) and \( b \in \mathbb{R}^{n} \); \( \mathbb{R} \) is real number field.

We know that for unique solution \( \det(A) \neq 0 \). Let us assume, without loss of generality, that none of the diagonal elements of \( A \) are zero; otherwise rows will be interchanged to make the diagonal element nonzero.

Now the coefficient matrix \( A \) can be decomposed as \( A = (D + U + L) \)

where \( D = (d_{ij}) \) is a diagonal matrix, \( L = (l_{ij}) \) is a strictly lower triangular matrix and \( U = (u_{ij}) \) is a strictly upper triangular matrix. Then according to classical Jacobi method, Eq.(1) can be rewritten as

\[ x^{(k+1)} = H_{j}x^{(k)} + V_{j} \]  

with \( x^{(k)} = \{x^{(k)}_1, x^{(k)}_2, \cdots, x^{(k)}_n\}^T; k = 0, 1, 2, \cdots \)

where \( H_{j} = D^{-1}(-L - U) \) is the Jacobi iteration matrix and \( V_{j} = D^{-1}b \).

On introducing SR technique (Engeln and Uhlig 1996, Gourdin and Boumahrat 1999) Eq. (2), can be written as

\[ x^{(k+1)} = H_{j(\omega)}x^{(k)} + V_{j(\omega)} \]  

where, \( H_{j(\omega)} = D^{-1}[(1-\omega)I - \omega(L + U)] \), \( V_{j(\omega)} = \omega D^{-1}b \), \( \omega \in (\omega_L, \omega_U) \) is the relaxation factor which influences the convergence rate greatly and \( I \) is the identity matrix; also \( \omega_L \) and \( \omega_U \) are the lower and upper bound of \( \omega \).

Similarly according to the classical Gauss-Seidel method by introducing SR technique (Engeln and Uhlig 1996, Gourdin and Boumahrat 1999) Eq. (1), can be written as

\[ x^{(k+1)} = H_{g(\omega)}x^{(k)} + V_{g(\omega)} \]  

Where \( H_{g(\omega)} = (1 + \omega D^{-1}L)^{-1}[(1-\omega)I - \omega D^{-1}U] \) is the Gauss-Seidel-SR iteration matrix and \( V_{g(\omega)} = \omega (1 + \omega D^{-1}L)^{-1}D^{-1}b \).

3. UNIFORM ADAPTIVE HYBRID EVOLUTIONARY ALGORITHMS

3.1 The Existing Hybrid Evolutionary Algorithms

The main aim of the hybridization of the classical SR methods with the evolutionary computation techniques is to self-adapt the relaxation factor used in the classical SR technique. The relaxation factors are adapted on the basis of the fitness of individuals (i.e. how well an individual solves the equations). Similar to many other evolutionary algorithms, the hybrid algorithm always maintains a population of approximate solutions of the linear equations. Each solution is represented by an individual. The initial population is generated randomly from the field \( \mathbb{R}^{n} \). Each relaxation factor associates with each different individual. Crossover in the hybrid algorithm involves all individuals in a population. If the population is of size \( N \), then the crossover will have \( N \) parents and generates \( N \) offspring through linear combination. Mutation is achieved by performing one iteration of classical (Gauss-Seidel or Jacobi) method with SR technique. The mutation is stochastic since \( \omega \) used in the iteration are initially determined between \( \omega_L (= 0) \) and \( \omega_U (= 2) \), is adapted stochastically in each generation. The fitness of an individual is evaluated on the basis of the error of an approximate solution. For example, given an approximate solution (i.e., an individual) \( z \), its error is defined by \( \|e(z)\| = \|Az - b\| \). The relaxation factors are adapted after each generation, depending on how well an individual performs (in terms of the error). The main steps of the existing JBUA and the GSBUA hybrid algorithms are – Initialization, Crossover, Mutation, Adaptation and Selection mechanism (Jun et al. 2000, Jamali et al. 2004). The pseudo-code structure of the existing hybrid evolutionary algorithm is given below:

**Algorithm** _JBUA/GBSUA(_)

begin
\[ I \leftarrow 0; \] /* Initialize the generation counter */
\[ X^{(0)} = \{x^{(0)}_1, x^{(0)}_2, \cdots, x^{(0)}_N\}; \] /* Initialize population*/
\[ /*Here x^{(t)}_i \Rightarrow i-th individual at t-th generation */


Randomly generate: \( \omega_i \in (0,2) \);

/* Initial relaxation factors */
Evaluate population: \( ||e(X)|| = ||e(z)|| : z \in X \);

While (not termination-condition) do
Select individuals for reproduction:

Apply operators:
  Crossover: \( X^{(k+c)} = R(X^{(k)}) \);
/* R is stochastic matrix & superscript c indicates Crossover */
Mutation: \( x_j^{(k+m)} = H_{q(o_j)}X^{(k+c)} + V_{q(o_j)} \);
/* where \( q \in \{j, g\}, "j" indicate Jacobi based method and "g" indicate Gauss-Seidel based method */
Evaluate newborn offspring:
\( ||e(X^{(k+m)})|| = ||e(\bar{X}^{(k+m)})|| : \bar{X}^{(k+m)} \in X^{(k+m)} \);
Adaptation of \( \omega \):
\( \omega_x = f_x(\omega_x, \omega_y, p_x) \) & \( \omega_y = f_y(\omega_x, \omega_y, p_y) \);
/* \( p_x \) and \( p_y \) are adaptive probability functions */
Selection and reproduction: \( X^{(k+1)} = \phi(X^{(k+m)}) \);
\( k \leftarrow k + 1 \);
/* Increase the generation counter */
end

As the adaptation and the selection are the main characteristic mechanisms of the existing hybrid algorithms (as well as proposed modified algorithm also), so we have described them in brief below:

Adaptation:
Let \( x^{(k+m)} \) and \( y^{(k+m)} \) be two offspring individuals with relaxation factors \( \omega_x \) and \( \omega_y \) and with errors (fitness values) \( ||e(x^m)|| \) and \( ||e(y^m)|| \) respectively. Then the relaxation factor \( \omega_x \) and \( \omega_y \) are adapted as follows:
(a) If \( ||e(x^m)|| > ||e(y^m)|| \),
(i) then \( \omega_x \) is moved toward \( \omega_y \) by setting
\[ \omega_x^{m} = (0.5 + p_x)(\omega_x + \omega_y) \] (5)
and (ii) \( \omega_y \) is moved away from \( \omega_x \) by setting
\[ \omega_y^{m} = \begin{cases} 
\omega_y + p_y(\omega_U - \omega_y), & \text{when } \omega_y > \omega_x \\
\omega_y + p_y(\omega_L - \omega_y), & \text{when } \omega_y < \omega_x 
\end{cases} \] (6)
where \( p_x \in (-0.01, 0.01) \) and \( p_y \in (0.008, 0.012) \) are the uniform adaptive (UA) probability parameters of \( \omega_x \) and \( \omega_y \) respectively. Note that \( \omega_x^{m} \) and \( \omega_y^{m} \) are adapted relaxation factors correspond to \( \omega_x \) and \( \omega_y \) respectively.
(b) If \( ||e(x^m)|| < ||e(y^m)|| \), then \( \omega_x \) and \( \omega_y \) are adapted in the same way as above but in the reverse order of \( \omega_x^{m} \) and \( \omega_y^{m} \).
(c) If \( ||e(x^m)|| = ||e(y^m)|| \), no adaptation will take place i.e.
\[ \omega_x^{m} = \omega_x \text{ and } \omega_y^{m} = \omega_y. \]

Selection and Reproduction:
At first algorithm select the best \( N/2 \) offspring individuals according to their fitness values (errors), then other \( N/2 \) offspring are reproduced (i.e. each parent individuals, which produce offspring containing in the best offspring set, generate two offspring). Thus the next generation of \( N \) individuals is formed.
3.2 The proposed modified hybrid evolutionary algorithms

The key idea behind the proposition of the modified algorithms (Modified Jacobi Based Uniform Adaptive (MJBUA) hybrid evolutionary algorithm and the Modified Gauss-Seidel Based Uniform Adaptive (MGSBUA) hybrid evolutionary algorithm) is to examine the necessity of crossover operation for solving set of linear equations. So the proposed modified hybrid evolutionary algorithms contains all steps of the JBUA and GSBUA hybrid evolutionary algorithms except the step – crossover. So we are not repeating the pseudo-code structures of the both modified uniform hybrid evolutionary algorithms here, as they will be same as that of JBUA and GSBUA except the crossover portion.

4. PERFORMANCE OF THE MODIFIED ALGORITHMS

In order to evaluate the effectiveness of the proposed MJBUA and MGSBUA hybrid algorithm, a number of numerical experiments have been carried out to solve the Eq. (1). The following settings are used for all the experiments:

Dimension of unknown variable, \( n = 200 \); Population size, \( N = 2 \). It is worthwhile to mention here that in the papers (Jamali et al., 2003; Jamali et al., 2004a; Jamali et al., 2004b; Jamali et al., 2004c) authors considered small population size, namely 2, and have gotten nice results in their experiments. Note that increasing of number of population implies increasing of computational effort. Moreover as, here we will mainly compare our proposed modified algorithms with corresponding algorithms proposed in paper. so we also consider population of size 2. We also like to inform that we try to consider same parameter setting as reported in paper if possible.

Boundary of relaxation factor, \((\omega_L, \omega_U) = (0, 2)\)

Initial domain from which each individual \(x\) of population \(X\) be initialized in \((-30, 30)\) and threshold error \(\eta = 10^{-7}\). Also the relaxation factors and the stochastic matrix \(R\) are generated randomly.

The first problem (problem \(P_1\) in Table 1 & 2) was set by considering\(a_{ij} \in (100, 200); a_{ij} \in (-10, 10); b_{ij} \in (100, 200), i, j = 1, \cdots, n\). A single set of parameters was generated randomly from the above-mentioned problem and the following two experiments were carried out. The problem was solved with an error smaller than the threshold error \(\eta = 10^{-7}\).

In the first experiment had been made for the comparison between the JBUA and the proposed MJBUA. Figure 1 shows the numerical results of this experiment. From this experiment, two important observations came out. Firstly the proposed MJBUA algorithm is comparable with the JBUA algorithm in terms of number of
generation (MJBUA algorithm is a bit faster though not very significant). Secondly the proposed MJBUA algorithm required less amount of time than that of JBUA algorithm (this is due to the absent of crossover step). In the second experiment, the comparison between the GSBUA and the proposed MGSBUA had been made. Figure 2 shows the numerical results of this experiment.

Table 1: Comparison between the JBUA and the proposed MJBUA hybrid algorithms for several randomly generated test problems

<table>
<thead>
<tr>
<th>Label of Test Problems</th>
<th>Domain of the elements of the coefficient matrix A &amp; the right side constant vector b of the test Problems</th>
<th>MJBUA ALG.</th>
<th>JBUA ALG.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Generation (Elapsed)</td>
<td>Elapse time (milli sec)</td>
</tr>
<tr>
<td>P₁</td>
<td>a₁∈(100,200); a₂∈(-10,10); b₁∈(100,200)</td>
<td>62</td>
<td>390</td>
</tr>
<tr>
<td>P₂</td>
<td>a₁∈(1,400); a₂∈(-4,4); b₁=100</td>
<td>171</td>
<td>1016</td>
</tr>
<tr>
<td>P₃</td>
<td>a₁∈(-50,50); a₂∈(-1,1); b₁∈(-1,1)</td>
<td>37</td>
<td>609</td>
</tr>
<tr>
<td>P₄</td>
<td>a₁∈100; a₂∈(-1,1); b₁∈(-100,100)</td>
<td>42</td>
<td>539</td>
</tr>
<tr>
<td>P₅</td>
<td>a₁=50; a₂∈(-10,10); b₁∈(-5,5)</td>
<td>10</td>
<td>156</td>
</tr>
<tr>
<td>P₆</td>
<td>a₁=50; a₂∈(-1,1); b₁=2</td>
<td>10</td>
<td>188</td>
</tr>
<tr>
<td>P₇</td>
<td>a₁=20i; a₂=(100-j)/20; b₁=10i</td>
<td>79</td>
<td>1282</td>
</tr>
<tr>
<td>P₈</td>
<td>a₁=20n; a₂=200; b₁=i</td>
<td>Not conv.</td>
<td>--</td>
</tr>
<tr>
<td>P₉</td>
<td>a₁=(-20,200); a₂=(-2,3); b₁=(-2,3)</td>
<td>475</td>
<td>7406</td>
</tr>
<tr>
<td>P₁₀</td>
<td>a₁=40; a₂=(-4,4); b₁=200</td>
<td>73</td>
<td>1170</td>
</tr>
<tr>
<td>P₁₁</td>
<td>a₁=(-50,50); a₂=(-1,1); b₁=(-1,1)</td>
<td>34</td>
<td>530</td>
</tr>
</tbody>
</table>

Table 2: Comparison between the GSBUA and the proposed MGSBUA hybrid algorithms for several randomly generated test problems

<table>
<thead>
<tr>
<th>Label of Test Problems</th>
<th>Domain of the elements of the coefficient matrix A &amp; the right side constant vector b of the test Problems</th>
<th>MGSBUA ALG.</th>
<th>GSBUA ALG.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Generation (Elapsed)</td>
<td>Elapse time (milli sec)</td>
</tr>
<tr>
<td>P₁</td>
<td>a₁∈(100,200); a₂∈(-10,10); b₁∈(100,200)</td>
<td>34</td>
<td>260</td>
</tr>
<tr>
<td>P₂</td>
<td>a₁∈(1,400); a₂∈(-4,4); b₁=100</td>
<td>92</td>
<td>710</td>
</tr>
<tr>
<td>P₃</td>
<td>a₁∈(-50,50); a₂∈(-1,1); b₁∈(-1,1)</td>
<td>44</td>
<td>625</td>
</tr>
<tr>
<td>P₄</td>
<td>a₁∈100; a₂∈(-1,1); b₁∈(-100,100)</td>
<td>40</td>
<td>495</td>
</tr>
<tr>
<td>P₅</td>
<td>a₁=50; a₂∈(-10,10); b₁∈(-5,5)</td>
<td>8</td>
<td>141</td>
</tr>
<tr>
<td>P₆</td>
<td>a₁=50; a₂∈(-1,1); b₁=2</td>
<td>13</td>
<td>204</td>
</tr>
<tr>
<td>P₇</td>
<td>a₁=20i; a₂=(100-j)/20; b₁=10i</td>
<td>112</td>
<td>1468</td>
</tr>
<tr>
<td>P₈</td>
<td>a₁=20n; a₂=i; b₁=i</td>
<td>60</td>
<td>859</td>
</tr>
<tr>
<td>P₉</td>
<td>a₁=(-20,200); a₂=(-2,3); b₁=(-2,3)</td>
<td>554</td>
<td>7141</td>
</tr>
<tr>
<td>P₁₀</td>
<td>a₁=40; a₂=(-4,4); b₁=200</td>
<td>107</td>
<td>1422</td>
</tr>
<tr>
<td>P₁₁</td>
<td>a₁=(-50,50); a₂=(-1,1); b₁=(-1,1)</td>
<td>13</td>
<td>111</td>
</tr>
</tbody>
</table>

Again two observations came out – (i) the proposed MGSBUA algorithm is comparable with the GSBUA algorithm in terms of number of generation (MGSBUA is a bit faster as usual but not significant) and (ii) the proposed MGSBUA algorithm required less amount of time than that of GSBUA algorithm (due to the absent of crossover step as usual).

Table 1 and 2 represent eleven test problems in each labeled Pᵢ; i = 1, 2 . . . 11, with dimension, n = 200. For each test problem Pᵢ, the coefficient matrix A and constant vector b all were generated uniformly and randomly within given domains. Table 1 shows the comparison between the numbers of generations (iterations) of the JBUA and the proposed MJBUA hybrid algorithms with respect to the threshold error, η = 10⁻⁷. One observation can be made immediately from this table that the proposed MJBUA hybrid algorithm is comparable with the existing JBUA hybrid algorithm for all the problems. Another observation is that the proposed MJBUA computationally is relatively cheaper. Again Table 2 shows the comparison between the numbers of generations
(iterations) of the GSBUA and the proposed MGSBUA hybrid algorithms with the threshold error, $\eta = 10^{-7}$. We again notice in the table that the proposed MGSBUA hybrid algorithm is comparable with the corresponding counterpart for all the problems and computationally cheaper as well. In order to study the effect of the dimension $n$ of the coefficient matrix $A$ on the modified hybrid evolutionary algorithms, we set problems with $a_{ii} = n$, $a_{ij} \in (-n/4, n/4)$, $b_{i} = n$; where the value of $n$ were set at 200, 100, 50, 25 and 10. For each value of $n$ the problem was generated randomly within proposed domain. Table 3 and 4 show the generation history of both the modified hybrid algorithms and their existing counterparts for the above problem. From the tables, it is observed that the effect of dimensions of the coefficient matrix on MGSBUA and its counterpart are similar as well as that of MJBUA and its counterpart are almost similar.

**Table 3**: Effect of dimension of coefficient matrix on both MGSBUA and GSBUA algorithms

<table>
<thead>
<tr>
<th>Dimension ($n$)</th>
<th>MGSBUA ALG. Generation (Elapsed)</th>
<th>MGSBUA ALG. Elapse time (mili sec)</th>
<th>GSBUA ALG. Generation (elapsed)</th>
<th>GSBUA ALG. Elapse time (mili sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>168</td>
<td>2200</td>
<td>169</td>
<td>2266</td>
</tr>
<tr>
<td>100</td>
<td>34</td>
<td>171</td>
<td>46</td>
<td>293</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>016</td>
<td>20</td>
<td>310</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>2</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 4**: Effect of dimension of coefficient matrix on both MJBUA and JBUA algorithms

<table>
<thead>
<tr>
<th>Value of $n$</th>
<th>MJBUA ALG. Generation (Elapsed)</th>
<th>MJBUA ALG. Elapse time (mili sec)</th>
<th>JBUA ALG. Generation (elapsed)</th>
<th>JBUA ALG. Elapse time (mili sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>162</td>
<td>2656</td>
<td>170</td>
<td>2828</td>
</tr>
<tr>
<td>100</td>
<td>32</td>
<td>140</td>
<td>31</td>
<td>156</td>
</tr>
<tr>
<td>50</td>
<td>19</td>
<td>15</td>
<td>19</td>
<td>31</td>
</tr>
<tr>
<td>25</td>
<td>15</td>
<td>1</td>
<td>15</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>4</td>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 3**: Curve (a) represents Jacobi-based hybrid generation history, curve (b) represents Classical Jacobi-SOR iteration history and curve (c) represents Classical Jacobi iteration history.

In order to justify the efficiency of the proposed hybrid evolutionary algorithm with respect to classical approaches, we performed another experiment (which was also discussed in Jamali et al., 2003).

Figure. 3 represents the decreasing of error, for the problem $P_{12}$: $a_{ii} = 2n$; $a_{ij} = i$; $b_{i} = i$ produced by the methods Jacobi-based hybrid algorithm with initial relaxation factors 1.0 and 1.5, Classical Jacobi-SR method with relaxation factor 1.5 and Classical Jacobi method (i.e. relaxation factor is 1.0). Note that the threshold error of this experiment was $\eta = 10^{-11}$. It is clear, from Figure 3, that Jacobi-based hybrid algorithm outperforms the Classical Jacobi-SR method as well as Classical Jacobi method. It was also showed by Jamali et al. (2003) that Classical SR based techniques are very sensitive with relaxation factors.
We would like to mention that a total of ten independent runs with different samples were conducted and the average results are reported here. Also all the experiments were performed in the same environment (1 GHz processor with 512 Mb RAM in Pentium IV PC).

5. **CONCLUDING REMARKS**

In this paper by omitting the crossover operation from existing GSBUA and JBUA (Jun et al. 2000, Jamali et al. 2004), the MJBUA and MGSBUA hybrid algorithms respectively has been proposed. The necessity of the crossover operation is investigated by comparing the performance of the proposed modified MJBUA and MGSBUA with that of the JBUA and the GSBUA hybrid algorithms respectively. The preliminary investigation has showed that both the proposed MJBUA and MGSBUA hybrid algorithms are comparable in terms of generation (no. of iteration) with the JBUA and GSBUA algorithms respectively. Also the both proposed MJBUA and MGSBUA hybrid algorithms relatively cheaper regarding computational cost. Furthermore since proposed modified hybrid algorithms have no crossover operation, so they require less memory allocation and less computational effort to solve the problems. Moreover, the proposed modified hybrid algorithms are also very simple and easier to implement both in sequential and parallel computing environments. It may thus conclude that for solving linear equations by uniform adaptive hybrid evolutionary algorithms, crossover is superfluous step. That is mutation (among the two evolutionary steps – mutation and crossover) is enough for solving linear equation by the uniform as well as time variant (Jamali et al., 2005) adaptive hybrid evolutionary algorithm.

**REFERENCES**


Hashem, M. M. A.: Global Optimization Through a New Class of Evolutionary Algorithm, Ph.D. dissertation, Saga University, Japan, pp 29-30, 1999


