https://doi.org/10.26682/sjuod.2022.25.2.4
Journal of University of Duhok., Vol. 25, No.2 (Pure Engineering Sciences), Pp 41-49, 2022

HYBRIDIZATION GRADIENT BASED METHODS WITH GENETIC ALGORITHM FOR SOLVING SYSTEMS OF LINEAR EQUATIONS

AYAD **R**AMADHAN **A**LI and **B**AYDA **G**HANIM **F**ATHI
Dept. of Mathematics, Faculty of Science, University of Zakho, Kurdistan Region-Iraq

(Received: April 24, 2022; Accepted for Publication: July 4, 2022)

ABSTRACT

In this paper, we propose two hybrid gradient based methods and genetic algorithm for solving systems of linear equations with fast convergence. The first proposed hybrid method is obtained by using the steepest descent method and the second one by the Cauchy-Barzilai-Borwein method. These algorithms are based on minimizing the residual of solution which has genetic characteristics. They are compared with the normal genetic algorithm and standard gradient based methods in order to show the accuracy and the convergence speed of them. Since the conjugate gradient method is recommended for solving large sparse and symmetric positive definite matrices, we also compare the numerical results of our proposed algorithms with this method. The numerical results demonstrate the robustness and efficiency of the proposed algorithms. Moreover, we observe that our hybridization of the CBB method and genetic algorithm gives more accurate results with faster convergence than other mentioned methods in all given cases.

***KEYWORDS*:** Genetic Algorithm, Hybrid Genetic Algorithm, Steepest Descent Method, Cauchy-Barzilai-Borwein Method, Systems of Linear Equations.

1. INTRODUCTION

There are various known iterative methods for solving a system of linear equations

$$Ax = b, \qquad (1.1)$$

where A is an $n \times n$ real matrix and x, b are $n \times 1$ real vectors. Genetic algorithm is a new developing research field of interest in finding solutions to these sets of equations.

A Genetic Algorithm is an evolutionary process based on Artificial Intelligence. It is a stochastic process or algorithm for solving optimization problems that are non-deterministic and non-mathematical. John Holland first proposed the concept of the genetic algorithm in 1975, with the goal of making computers mimic nature. He was fascinated with algorithms that manipulate binary digit strings to solve a problem that resembled natural evolution, i.e., designing an algorithm that is an abstract of natural development. Holland got the idea for the genetic algorithm from Charles Darwin's evolutionary theory (1859) called Darwinian evolution. The Darwinian evolutionary concept of "survival of the fittest" declared that only species that are fitted to survive may reproduce their kind. The evolutionary process is the computer matching of Charles Darwin that yields a genetic algorithm (Holland, 1975).

In a variety of fields, such as science, social sciences, and engineering, systems of linear equations are used to solve problems. Several well-known traditional techniques for solving systems of linear equations based on theoretical principles occur. The use of genetic algorithm to find a solution to this set of equations is a new and emerging study subject of interest. This algorithm is based on the notion of solution evolution, in which generations of solutions are produced stochastically using a specified fitness function to find the best fit solution to the situation. The genetic algorithm has been used to solve a range of scientific problems, including scheduling, timetabling, and the problems of traveling salesmen. It has been effectively used in many optimization problems, but its application for solving systems of equations is still being researched (Ikotun Abiodun, Lawal Olawale, & Adelokun Adebowale, 2011).

In this paper, two-hybrid genetic algorithms have been proposed for solving symmetric linear systems with positive definite coefficient matrices and nonlinear systems of equations. Some of their matrices are ill-conditioned and high dimensions. The proposed hybrid algorithms are based on a normal evolutionary

ayad.ali@staff.uoz.edu.krd; Bayda.fathi@uoz.edu.krd

algorithm and gradient search techniques. The best individual of the population propagates to the next generation using gradient information. We use the best individual to accelerate the convergence of the SD (Cauchy, 1847) and the CBB methods (Raydan & Svaiter, 2002).

This paper has the following structure: In section two, the problem statement for our work is stated, in section three, some related works are presented. In section four, the fundamental steps of the genetic algorithm are shown. We introduce two suggestion optimization techniques for solving systems of linear equations in section five. In section six, we propose two hybrid genetic algorithms. In section seven the numerical results and discussions are given. Finally, we conclude in section eight.

2. PROBLEM STATEMENT

Systems of linear equations are used to solve linear problems in a variety of topics, including science, the social sciences, and engineering. There are a number of well-known conventional methods for solving systems of linear equations that are founded on theoretical ideas. A novel and developing area of research that is of interest is the use of genetic algorithms to solve this system of equations. This algorithm is based on the idea of solution evolution, where successive generations of solutions are generated stochastically while using a designated fitness function to discover the solution that best fits the circumstance. Scheduling, timetabling, and issues with traveling salespeople are just a few of the scientific issues that the genetic algorithm has been used to solve. Although many optimization problems have successfully been solved using it, its applicability in solving systems of equations is currently under investigation [39].

This research focuses on investigating how quickly gradient algorithms can solve generic linear systems of equations. Additionally, we try to suggest methods to combine the continuous method (numerical optimization) with the discrete technique (genetic algorithm) to obtain the overall solution. In short, our aim is to provide some general-purpose algorithms for solving linear systems of equations.

We will use two known gradient-type approaches to solve a linear system of equations in our study. Numerous applications, such as mechanical structure design, medical imaging, difficult combinatorial problems, etc., might include these kinds of problem.

This feature denotes quick convergence in terms of the number of iterations (computations of the approximate solutions). This makes it possible to obtain high-accuracy solutions. All known polynomial time techniques, however, have a common drawback: the computing cost each iteration rises nonlinearly with the problem's design dimension.

3. RELATED WORKS

M. Ikotun Abiodun et al. examined the origins of the Genetic Algorithm and its usefulness in solving systems of simultaneous equations in 2011. To solve seven distinct systems of simultaneous linear equations, they used the Genetic Algorithm on the one hand and the Gaussian elimination approach on the other. They then compared the results of the two approaches. The Genetic Algorithm was shown to be extremely successful in discovering all feasible sets of solutions for any given system of simultaneous linear equations.(Ikotun Abiodun et al., 2011).

A general introduction to the Genetic Algorithm is given, as well as its application in solving Systems of Linear Equations and the implications of changing the Population Size and Number of Generations. The genetic algorithm simultaneous linear equation solver software was ran numerous times with different sets of simultaneous linear equations and varied population sizes and generations to see how they affected solution creation. It was discovered that a small population size does not provide perfect answers as quickly as a big population size does, and that the number of generations, whether small or large, had no influence on the achievement of a perfect solution(Ikotun, Akinwale, & Arogundade, 2016).

To identify the Fittest Chromosomes, an efficient genetic method was used to a linear programming issue. The experimental results identify the best chromosomes for a constraint linear programming problem, resulting in a better solution(Datta, 2012).

Punam S Mhetre used the Genetic Algorithm as a nonlinear approach to solve linear and nonlinear equation systems in 2012, with the goal of determining the key benefits received as a consequence of employing GA(Mhetre, 2012).

Lubna Zaghlul Bashir solved a linear equation problem using genetic algorithms in 2015(Bashir, 2015).

Some hybrid genetic algorithms have been developed by combining genetic algorithms with gradient-based heuristic search strategies, such as conjugate gradient, and have recently been used to solve hard scientific problems(Sun & Zhang, 2006).

In 2007, Tahk et al. introduced a hybrid optimization approach for optimization with continuous parameters that combines evolutionary algorithms with gradient search techniques(Tahk, Woo, & Park, 2007).

Okamoto et al. introduced a hybrid genetic algorithm for non-linear numerical optimization in 1998, which included the modified Powell technique(Okamoto, Nonaka, Ochiai, & Tominaga, 1998).

Chelouah and Patrick introduced a continuous new hybrid approach for continuous multimodal optimization problems in 2003, integrating genetic algorithm with Nelder-Mead simplex algorithm(Chelouah & Siarry, 2003) .

In computer-aided design, genetic algorithms are used. Design is a difficult technical process that is becoming more computer-aided. The design task is frequently viewed as an optimization problem, with parameters or structures characterizing the highest-quality design algorithms and several methods in which they might tackle tough design challenges. They addressed a number of sophisticated genetic algorithms that have been shown to be effective in tackling complex design issues(El-Emary & Abd El-Kareem, 2008; Renner, 2004).

For all works mentioned above, some researchers used hybrid genetic algorithms with gradient optimization techniques to solve nonlinear equations, and others used only genetic algorithms to solve linear systems. However, these techniques do not work well for ill-conditioned systems or solving systems with large dimensions. The solution of a linear system is equivalent to the minimization of a nonlinear quadratic function, but what if the system is large and ill-conditioned? In this case, we aim to propose two hybrid genetic algorithms with common gradient optimization methods that can solve systems for large dimensions and ill-conditioned systems with fast convergence.

4. GENETIC ALGORITHM (GA)

In most cases, a genetic algorithm is made up of two processes. The first process is the selection of suitable parents for the next generation(Karakatič & Podgorelec, 2015; Saragih & Nababan, 2019) The second step is reproduction, which involves modifying the fitness value of the new generation of children by using crossover and mutation operators on the selected parents(Razali & Geraghty, 2011).

The flowchart of the GA is presented in Fig.1 as was shown by Pal & Parashar in 2014, where its steps are written in the following:

4.1 Genetic Algorithm (Hossain et al., 2019)

The typical steps to be followed in designing genetic algorithm are:

Step 1: [Start][initialization] Randomly create an initial population of n chromosomes (i.e. appropriate solutions to the problem)

Step 2: [Fitness] For each chromosome, the fitness function is calculated.

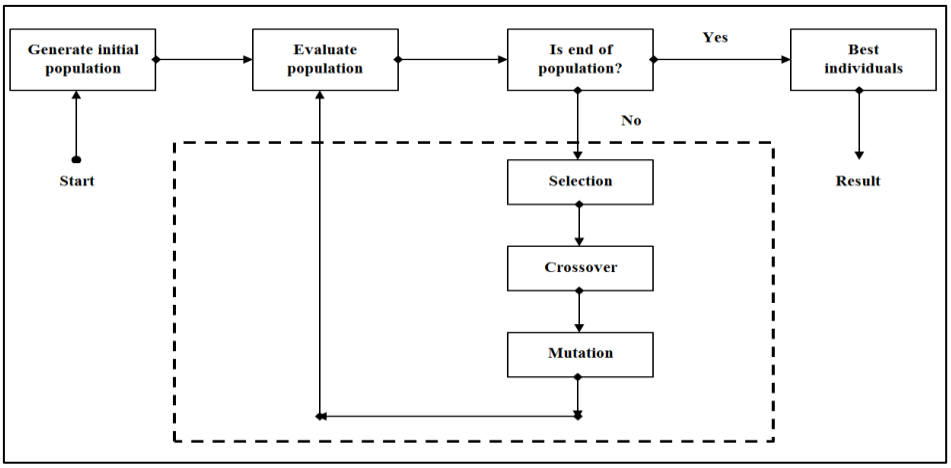
Step 3: [New population] To make a new population, repeat the instructions below.

- **[Selection]** The present population is utilized to determine which chromosome pairs should be mated. The fitness probability of parent chromosomes is used to choose them. Highly fitting chromosomes are more likely to be selected for mating.
- **[Crossover]** The parents will crossover to create new offspring if their crossover chance is high. If there is no crossover, the offspring will be the same as the parents
- **[Mutation]** Create new offspring with a probability of a mutation at each chromosome position.
- **[Acceptance]** The chromosomes of the generated offspring are placed in the new population.

Step 4: [Replacement] Use the generated population for a subsequent run of the program.

Step 5: [Test] If the end condition is reached, stop and return the best solution in the current population.

Step 7: [Loop] Continue to the second step.



Fig(1):- Flowchart of the genetic algorithm (Pal & Parashar, 2014).

5. GRADIENT BASED METHODS

5.1 The Steepest Descent (SD) Method

The steepest descent (SD) method is the analog of finding the minima of a function by following the derivative in higher dimensions. In 1847, this method was first proposed by Cauchy (Cauchy, 1847). The solution of $Ax = b$ is the point in R^n where the quadratic form $f(x) = \frac{1}{2}x^T Ax - b^T x + c$ achieves the minimum value. This can be verified by expanding $f(x)$ and setting the partial derivatives to be 0. The positive semidefinite constraint ensures that the solution of $Ax = b$ is a local minimum for $f(x)$ and not a saddle point.

The steepest descent method is an iterative method that starts at an arbitrary point x_0 and obtains x_{k+1} from x_k by moving in the direction r_k reverse to the gradient. The gradient is the direction of steepest increase for the value of $f(x)$ locally, the goal is to minimize $f(x)$ so we move in a direction opposite to the gradient. The update rule is,

$$x_{k+1} = x_k - \alpha_k r_k \qquad (5.1)$$

Here $r_k = Ax_k - b$ is the direction opposite to the gradient and we choose α_k such that $r_k^T \cdot r_{k+1} = 0$. The geometric intuition for the choice of α is the following: As we move in the direction r_k opposite to the gradient, the gradient starts changing, as long as the projection of the gradient onto the direction r_k is negative the value of the objective function decreases. The transition point occurs when the new gradient is orthogonal to the current direction.

The value of α corresponding to the orthogonality condition can be calculated easily with some algebra(Fathi, 2013),

$$r_{k+1} = Ax_{k+1} - b \qquad (5.2)$$

$$r_k^T \cdot r_{k+1} = 0 \Rightarrow \alpha_k = \frac{r_k^T r_k}{r_k^T A r_k} . \qquad (5.3)$$

5.2 Outline of the SD Method (Cauchy, 1847):

Step1: Given a matrix $A \in \mathbb{R}^{(n\times n)}, b \in \mathbb{R}^n$ and an initial approximation $x_0 \in \mathbb{R}^n$ to compute the solution $x = A^{-1}b$ of $Ax = b$.
Step 2: Set $r_0 = Ax_0 - b$.
Step 3: for $k = 0, 1, \ldots V$, where V is the maximum number of iterations.
Step 4: Set the step size $\alpha_k = \frac{r_k^T r_k}{r_k^T A r_k}$.
Step 5: Compute $x_{k+1} = x_k - \alpha_k r_k$.
Step 6: Compute the new residual defined by $r_{k+1} = Ax_{k+1} - b$.
Step 7: Until $\|r_{k+1}\|^2$ is zero or small enough.

5.3The Cauchy-Barzilai-Borwein (CBB) Method

Barzilai-Borwein-like methods have been applied in a wide range of applications due to their efficiency, simplicity, and minimal memory needs. The Cauchy-Barzilai-Borwein (CBB) algorithm is a modification of the Barzilai-Borwein (BB) algorithm (Raydan & Svaiter, 2002). Each iteration of this algorithm can be thought of as two iterations of the steepest descent method, where the step size only needs to be computed once but is used twice(Fathi, 2013). The algorithm has been defined as follows:

Let $x_0, y_0 \in \mathbb{R}^n$ and

$$y_k = x_k - \frac{(r_k, r_k)}{(Ar_k, r_k)} r_k, \tag{5.4}$$

where $r_k = Ay_k - b$. If

$$x_{k+1} = y_k - \frac{(r_k, r_k)}{(Ar_k, r_k)} r_k . \tag{5.5}$$

Then by using

$$Ay_k - b = r_k - \frac{(r_k, r_k)}{(Ar_k, r_k)} Ar_k \tag{5.6}$$

it can obtain

$$x_{k+1} = x_k - 2\frac{(r_k, r_k)}{(Ar_k, r_k)} r_k + \left(\frac{(r_k, r_k)}{(Ar_k, r_k)}\right)^2 Ar_k . \tag{5.7}$$

5.4Outline of the CBB Method (Raydan & Svaiter, 2002):

Step1: Given a matrix $A \in \mathbb{R}^{(n\times n)}, b \in \mathbb{R}^n$ and an initial approximation $x_0 \in \mathbb{R}^n$ to solve $Ax = b$.

Step 2: Set the initial residual $r_0 = Ax_0 - b$.

Step 3: for $k = 0, 1, \dots, V$, where V is the maximum number of iterations.

Step 4: Set the step size $\alpha_k = \frac{r_k^T r_k}{r_k^T A r_k}$.

Step 5: Compute $x_{k+1} = x_k - 2\alpha_k r_k + \alpha_k^2 A r_k$.

Step 6: Compute the new residual $r_{k+1} = Ax_{k+1} - b$.

Step 7: Stop as $\|r_{k+1}\|^2$ is zero or small enough.

Raydan and Svaiter demonstrated that in the elliptic norm, this approach converges Q-linearly. Furthermore, they discovered a unique property of this algorithm: increasing the condition number reduced the average number of iterations required by CBB(Raydan & Svaiter, 2002).

For solving systems of linear equations, the CBB approach has been shown to be clearly superior than the SD and BB methods in several numerical studies(Fathi, 2013).

6. TWO PROPOSED HYBRID GENETIC ALGORITHMS

This section contains two hybrid genetic algorithms for solving systems of linear equations.

6.1 First Proposed Hybrid Genetic (HGASD) Algorithm

The first proposed algorithm is a hybrid genetic algorithm with the steepest descent method and the steps of the first proposed hybrid genetic algorithm are:

1-Parameters:

Given a matrix $A \in \mathbb{R}^{(n\times n)}, b \in \mathbb{R}^n$, determine the number of individuals, generation (iteration), mutation rate, crossover rate value, and minimum fitness value (tolerance).

2-Create Initial Population:

The process begins with a population, which is a set of individuals. Each individual is a solution to the problem under consideration. In this work, the population number is equal to 100 which are decimals.

3- Fitness Evaluation:

The fitness function specifies how an individual should be fitted (i.e. the ability of an individual to compete with other individuals). It assigns a fitness value to each individual. The fitness value decides whether or not an individual will be selected for reproduction. The fitness function in this work is $f(x) = \|Ax - b\|^2$.

4- Selection Operator:

The purpose of the selection phase is to choose the fittest individuals (parents) and let them pass on their genes to the next generation. In this work we use tournament selection method, Tournament selection involves running several "tournaments" among a few individuals (chromosomes) chosen at random from the

population. The winner of each tournament (the one with the best fitness) is selected for crossover.

5 - Crossover Operator:

Crossover is usually applied to selected pairs of parents. Single point crossover is the most simple crossover operator, where a crossover point is selected randomly, and two-parent chromosomes are exchanged at this point. The crossover rate (CR) is the parameter that determines how many chromosomes are expected to undergo the crossover operation and it is taken equal to 0.98 in this work. All chromosomes in the population will be included in the crossover procedure when the crossover rate is set to 1.

6 – Mutation Operator:

The most common way of implementing mutation is the uniform mutation (which is used in this work); in uniform mutation one position is chosen randomly in the chromosome and changes its value with a probability.

7 - Hybridization step:

After mutation, a new population has been created. Each chromosome in the new population is considered as the initial point for the steepest descent method. The equations (5.1) and (5.3) are executed to generate a new population to start the genetic algorithm again until the stopping criterion is satisfied.

8 - Stop Criterion:

The stopping criterion decides whether the algorithm continues in searching or stops. In this work, the stop criterion depends on two approaches: either the number of generations reached or the fitness value found to be less than 10^{-6}.

6.2 Outlines of Our HGASD Algorithm:

Step1: Given a matrix $A \in \mathbb{R}^{(n\times n)}, b \in \mathbb{R}^n$ and tolerance $= 10^{-6}$.

Step2: Create the initial population randomly, and $k = 0, 1, \dots, V$, where V is the maximum number of generations (iterations).

Step3: Calculate fitness function for each chromosome in the population.

Step4: [New population] Create a new population by repeating the following steps until a new population is complete.

- **[Selection]** Select two parent chromosomes from a population according to their fitness.
- **[(Crossover) (recombination)]** Crossover the parents to form a new offspring (children).
- **[Mutation]** Mutate new offspring by using order change at a random position.

Step5: Let $x_k =$ new population.

Step6: Set the residual vector $r_k = Ax_k - b$.

Step7: Evaluate the step size α_k by (5.3).

Step8: Compute x_{k+1} by (5.1).

Step9: Calculate fitness function for each chromosome in the population, and then find minimum fitness value.

Step10: If $k = V$ or minimum fitness value is zero or less than the tolerance then stop; print minimum fitness value and solution x_{k+1}, otherwise, continue.

Step11: Set $k = k + 1$, go to step 3.

6.3 Second Proposed Hybrid Genetic (HGACBB) Algorithm

The Cauchy-Barzilai-Borwein (CBB) method is used in the second proposed hybrid algorithm and its steps as the first proposed algorithm except for the hybridization steps. In the hybridization step for this algorithm is after mutation, a new population has been created. Each chromosome in the new population is the initial point for the Cauchy-Barzilai-Borwein (CBB), and then the hybridization with CBB starts. The two main steps of the CBB method, (5.3) and (5.7), are executed to generate a new population to restart the genetic algorithm until the stopping criterion is satisfied.

The outline of our HGACBB algorithm is the same as the HGASD algorithm except for step 8 in which x_{k+1} is computed by using the equation (5.7).

7. NUMERICAL RESULTS AND DISCUSSION

In this work, the overall efficiency of the proposed methods is tested with numerical examples. The programs of all algorithms under consideration are written in MATLAB 2014 with double precision to introduce the computational performance of them. In all programs, the parameters of the GA and the proposed hybrid algorithms are: population numbers = 100, crossover rate = 0.98 and, mutation rate = 0.01. Stopping criterion for the gradient based methods is $\|r_k\|^2 \leq 10^{-6}$ and it is the minimum fitness value (tolerance) $\leq 10^{-6}$ for the normal GA and our hybrid algorithms (HGASD and HGACBB).

To investigate the applicability of the two proposed hybrid algorithms to solving system of linear equations, several systems of linear equations (positive integer only) of two different dimensions, 3×3 and 4×4, most of these systems are taken from (Ikotun Abiodun et al.,

2011), are used and their results are illustrated in Tables 1 and 2.

Table 1 gives the solutions of three systems of linear equations throw 20 iterations by using the methods of SD, CBB, GA, CG (H/S), HGASD and HGACBB. These solutions are compared with the exact solution to see the accuracy of the performance of all the considering algorithms in this work.

Table (1): Comparison of the standard SD method, CBB method, CG (H/S) method and GA with two proposed hybrid genetic algorithms (HGASD and HGACBB) for solving some given examples

Test No.	Equations	ρ	SD	CBB	CG(H/S)	GA	HGASD	HGACBB	Exact
1	$x_1+ 2x_2 +3x_3 = 3$ $x_1+2x_2 + x_3 = 6$ $x_1+ x_2 + 3x_3 = 7$	20.457	$x_1 = -270.1$ $x_2 = 98.638$ $x_3 = 59.321$	$x_1 = 15.5$ $x_2 = -4$ $x_3 = -1.5$	$x_1 = 15.083$ $x_2 = -4.607$ $x_3 = -1.471$	$x_1 = 3.0734$ $x_2 = 0.3766$ $x_3 = 0.6016$	$x_1 = 15.5$ $x_2 = -4$ $x_3 = -1.5$	$x_1 = 15.5$ $x_2 = -4$ $x_3 = -1.5$	$x_1 = 15.5$ $x_2 = -4$ $x_3 = -1.5$
2	$-10x_1+ 7x_2 - 5x_3 = 60$ $x_1 + x_3 = 26$ $-3x_1 +7x_2 +5x_3 = 14$	73.146	Fail	Fail	Fail	$x_1 = 6.4583$ $x_2 = 11.127$ $x_3 = -8.396$	$x_1 = 102.0009$ $x_2 = 100.0009$ $x_3 = -76.0007$	$x_1 = 102.0003$ $x_2 = 100.0003$ $x_3 = -76.0002$	$x_1 = 102$ $x_2 = 100$ $x_3 = -76$
3	$-x_1+2x_2+3x_3+16x_4= -33$ $x_1 +2x_2 + x_3 - 8x_4= 0$ $10x_1 +x_2+3x_3+5x_4= -7$ $x_1 - 2x_2 + 4x_3 - 9x_4= 1$	6.9815	Fail	Fail	Fail	$x_1 = 0.9887$ $x_2 = -1.5766$ $x_3 = -3.4088$ $x_4 = -1.0939$	$x_1 = 1.1374$ $x_2 = -2.6975$ $x_3 = -3.5904$ $x_4 = -0.9810$	$x_1 = 1.1370$ $x_2 = -2.6969$ $x_3 = -3.5902$ $x_4 = -0.9809$	$x_1 = 1.1374$ $x_2 = -2.6976$ $x_3 = -3.5904$ $x_4 = -0.9810$

Table2 shows the convergence speed of all algorithms for seven examples (Ikotun Abiodun et al., 2011). The speed of the convergence is based on generation (iteration) numbers. In this table, the convergence speed of the HGASD and HGACBB is compared with the normal GA, the SD method, the CBB method and the CG (H/S) method.

Table (2): Comparison of the standard SD method, the CBB method, the CG (H/S) method and GA with the two proposed hybrid algorithms (HGASD and HGACBB) for some tested systems of linear equations (Ikotun Abiodun et al., 2011).

Test No.	Equations	ρ	Number of generations(iterations)					
			SD	CBB	CG (H/S)	HGASD	HGACBB	Original GA
1	x1 + 2x2 + 3x3 = 14 x1 + x2 + x3 = 6 3x1 + 2x2 + x3 = 10	2.8416e+016	8	3	2	3	3	54
2	2x1 + 4x2 + x3 = 5 4x1 + 4x2 + 3x3 = 8 4x1 + 8x2 + x3 = 9	42.8811	Fail	12	79	11	7	24
3	10x1 + x2 + x3 = 12 2x1 + 10x2 + x3 = 13 2x1 + 2x2 + 10x3 = 14	1.5322	6	3	5	4	2	33
4	x1 + 2x2 + 3x3 = 6 2x1 + 4x2 + x3 = 7 3x1 + 2x2 + 9x3 = 14	21.9933	Fail	8	102	4	4	25
5	2x1 + x2 + 3x3 = 13 x1 + 5x2 + x3 = 14 3x1 + x2 + 4x3 = 17	41.0554	Fail	6	3	12	3	2194
6	2x1 + 4x2 + 8x3 = 44 4x1 + 6x2 + 10x3 = 66 6x1 + 8x2 + 10x3 = 84	128.4097	Fail	15	Fail	104	18	3500
7	4x1 + 3x2 + 2x3 + x4 = 10 3x1 + 2x2 + x3 + 4x4 = 9 2x1 + x2 + 4x3 + 3x4 = 14 x1 + 4x2 + 3x3 + 2x4 = 10	5.0000	Fail	17	4	11	7	1471

In order to test the convergence speed of our proposed algorithms, we compare their performance with other related algorithms using systems of linear equations with random positive definite matrices $A_{n\times n}$ and random vectors $b_{n\times 1}$, with different sizes ($n = 10,\ 50,\ 100, 150$) and condition number, ρ=40. All methods were run 500 times each. The results are demonstrated as in Tables 3 and 4.

Table 3 observes shows the convergence speed of the HGASD, HGACBB, the normal GA, the SD method, the CBB method and the CG (H/S) method based on generation numbers of the linear systems with symmetric positive definitive matrices (SPDM) (Fathi, 2013).

Table 4 tests the convergence speed based on generation(iteration) numbers of the random linear systems of equations with slightly non-symmetric positive definitive matrix (SNSPDM) (Fathi, 2013).

Table (3): Comparison of the standard SD method, CBB method, CG (H/S) method and GA with two proposed hybrid genetic algorithms (HGASD and HGACBB) for linear systems with SPDM

Test No.	Matrix Dimension	Number of generations(iterations)					
		SD	CBB	CG(H/S)	HGASD	HGACBB	Original GA
1	$A_{10\times 10}$	133	17	10	10	8	Over 2000
2	$A_{50\times 50}$	139	18	21	125	13	Over 2000
3	$A_{100\times 100}$	137	19	20	51	16	Over 2000
4	$A_{150\times 150}$	129	18	21	49	13	Over 2000

Tables 3 and 4 are shown that the convergence speed for our hybrid algorithms is better than for the standard gradient methods and the genetic algorithm without a hybrid. Moreover, the HGACBB for solving both types of systems, i.e. SPDM and NSPDM, is faster than all the other mentioned algorithms.

Table (4): Comparison of the standard SD method, CBB method, CG (H/S) method and GA with two proposed hybrid genetic algorithms (HGASD and HGACBB) for linear systems with SNSPDM

Test No.	Matrix Dimension	Number of generations(iterations)					
		SD	CBB	CG(H/S)	HGASD	HGACBB	Original GA
1	$A_{10\times 10}$	133	16	20	18	8	1500
2	$A_{50\times 50}$	107	20	27	26	11	Over 2000
3	$A_{100\times 100}$	141	21	31	51	21	Over 2000
4	$A_{150\times 150}$	125	24	32	62	17	Over 2000

Form the tables; it observes that our hybrid methods show better performance than the one without hybrid for all presented test problems. For linear systems with symmetric positive definite matrices, the best result we can obtain by using the CG-H/S method which is competitive to the results of the HGACBB when the condition numbers are small enough. If the condition number of the matrices are increased (ill- problems), then our proposed hybrid algorithms give superior results than the methods of SD, CBB, GA and even CG-H/S.

In general, for non-symmetric positive definite matrices, the HGASD and HGACBB are preferable to use rather than the GA and the standard gradient based methods.

By this, we can conclude that hybrid genetic algorithms are effective and more efficient than the normal genetic algorithm of solving symmetric systems of equations with positive definite coefficient matrices. The HGACBB is outperformance than all other algorithms for each case as reported in every table.

8. CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

In this paper, we proposed two hybrid genetic algorithms for solving systems of linear equations where their matrices are square and real. The numerical experiments showed that the hybrid genetic algorithm is more effective and efficient than the standard algorithms and the original genetic algorithm for solving all given problems. Although the CG method is better than HGASD for some cases, it gets worse than HGACBB for all cases. Therefore, we can notice that the idea of hybridization of the gradient method and genetic algorithm give faster and

ayad.ali@staff.uoz.edu.krd; Bayda.fathi@uoz.edu.krd

more accurate results than CG method even for systems of linear equations with symmetric positive definite matrix when the condition number increases.

This work gives motivation to use the idea of the hybridization the gradient type methods and genetic algorithm to solve more complicated problems such as the systems of linear equations with higher dimensions and non-square matrices moreover for general systems of nonlinear equations.

REFERENCES

Bashir, L. Z. (2015). Solve simple linear equation using evolutionary algorithm. World Scientific News, No. 19, 148–167.[1]

Cauchy, A. (1847). Méthode générale pour la résolution des systèms d'équations simultanées. Comput. Rend. Sci. Paris, vol. 25 , 536–538.[2]

Chelouah, R., & Siarry, P. (2003). Genetic and Nelder–Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. European Journal of Operational Research, vol. 148 ,No. 2, 335–348. https://doi.org/10.1016/S0377-2217(02)00401-0[3]

Datta, S. (2012). Efficient genetic algorithm on linear programming problem for fittest chromosomes. Journal of Global Research in Computer Science, vol. 3 ,No. 6, 1–7.[4]

El-Emary, I. M. M., & Abd El-Kareem, M. M. (2008). Towards using genetic algorithm for solving nonlinear equation systems. World Applied Sciences Journal, vol. 5 ,No. 3, 282–289.[5]

Fathi, G. B. (2013). *Gradient Optimization Algorithms with Fast Convergence*. Cardiff University, UK.[6]

Holland, J. H. (1975). Adaptation in natural and artificial systems, univ. of mich. press. Ann Arbor.[7]

Hossain, M., Tanim, A., Choudhury, S., Hayat, S., Kabir, M. N., & Islam, M. M. (2019). An Efficient Solution to Travelling Salesman Problem using Genetic Algorithm with Modified Crossover Operator. EMITTER International Journal of Engineering Technology, vol. 7 ,.[8]

Ikotun, A. M., Akinwale, A. T., & Arogundade, O. T. (2016). Parameter Variation For Linear Equation Solver Using Genetic Algorithm. Journal of Natural Sciences Engineering and Technology, vol. 15 ,No. 2, 42–50.[9]

Ikotun Abiodun, M., Lawal Olawale, N., & Adelokun Adebowale, P. (2011). The effectiveness of genetic algorithm in solving simultaneous equations. International Journal of Computer Applications, vol. 975 , 8887.[10]

Karakatič, S., & Podgorelec, V. (2015). A Survey of Genetic Algorithms for Solving Multi Depot Vehicle Routing Problem. Appl. Soft Comput., vol. 27 ,No. C, 519–532.[11]

Mhetre, P. S. (2012). Genetic algorithm for linear and nonlinear equation. International Journal of Advanced Engineering Technology, vol. 3 ,No. 2, 114–118.[12]

Okamoto, M., Nonaka, T., Ochiai, S., & Tominaga, D. (1998). Nonlinear numerical optimization with use of a hybrid genetic algorithm incorporating the modified Powell method. Applied Mathematics and Computation, vol. 91 ,No. 1, 63–72.[13]

Pal, D., & Parashar, A. (2014). Improved genetic algorithm for intrusion detection system. In *2014 International Conference on Computational Intelligence and Communication Networks* (pp. 835–839). IEEE.[14]

Raydan, M., & Svaiter, B. F. (2002). Relaxed steepest descent and Cauchy-Barzilai-Borwein method. Computational Optimization and Applications, vol. 21 ,No. 2, 155–167.[15]

Razali, N. M., & Geraghty, J. (2011). Genetic algorithm performance with different selection strategies in solving TSP. In *Proceedings of the world congress on engineering* (Vol. 2, pp. 1–6). International Association of Engineers Hong Kong, China.[16]

Renner, G. (2004). Genetic algorithms in computer-aided design. Computer-Aided Design and Applications, vol. 1 ,No. 1–4, 691–700.[17]

Saragih, R. I. E., & Nababan, D. (2019). Increase Performance Genetic Algorithm In Matching System By Setting GA Parameter. In *Journal of Physics: Conference Series* (Vol. 1175, p. 12100). IOP Publishing.[18]

Sun, L., & Zhang, W. (2006). An accelerated micro genetic algorithm for numerical optimization. In *Asia-Pacific Conference on Simulated Evolution and Learning* (pp. 277–283). Springer.[19]

Tahk, M., Woo, H., & Park, M. (2007). A Hybrid Optimization Algorithm of Evolutionary Algorithm and Gradient Search. Engineering Optimization, vol. 39 ,No. 1.[20]