The Use of Original and Hybrid Grey Wolf Optimizer in Estimating the Parameters of Software Reliability Growth Models

Jamal Salahaldeen Majeed Alneamy, PhD
Software Engineering Department,
Computer and Mathematics Science College,
University of Mosul, Iraq

Marwah Marwan Abdulazeez Dabdoob
Software Engineering Department,
Computer and Mathematics Science College,
University of Mosul, Iraq

ABSTRACT

In order to optimize the use of programs, it has become necessary to focus on issues like software reliability. In this work, the parameters of Software Reliability Growth Models (SRGMs) were estimated in depending on failure data and Swarm Intelligence, namely, Grey Wolf Optimizer (GWO). Then, the (GWO) was hybrid with Real Coded Genetic Algorithm (RGA) to obtain Hybrid GWO (HGWO).

The results that obtained from (GWO) are compared to the results of five algorithms: Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), the Dichotomous Artificial Bee Colony (DABC), Classic Genetic Algorithm (CGA) and the Modified Genetic Algorithm (MGA).

The results showed that (GWO) outperformed the rest of the algorithms in parameters estimating accuracy and performance using identical datasets. Sometimes, the (DABC) showed better performance than (GWO).

Other comparisons were made between (GWO) and (HGWO) and the results show that the hybrid algorithm outperformed the original one.

General Terms

Swarm Intelligence.

Keywords

Genetic algorithms, Grey Wolf optimizer, Software Reliability Growth Models .

1. INTRODUCTION

Software reliability is a key attribute to software quality. Reliability can be defined as "how well software meets its requirements" and also "the probability that the software will operate without failure for the specified period of time in a specified environment" [1]. Those software failures are introduced during different stages of software development life cycle by the system analysts, designers, programmers and managers [2].

The presence of a fault in a system may lead to a system failure, a failure causes the system performance to diverge from the specified performance. A fault (also called a defect) is an erroneous state of the system. There are no fixed definitions of a fault because they are different from different systems and in different situations, but in general, a fault can be defined as an existing portion in the system which can be removed by correcting the erroneous portion of the system [3]. Defect detection is usually a failure during a test phase; test software may discover a defect and the test continue its operation [4].

One of the hard issues is to develop reliable software specifically when the software modules are interdepending on each other and a lot of current software has this interdependency. Another hard issue is to know whether the delivered software to customers is reliable or not. The reliability can be known to software vendors after the software is delivered, the vendors receive customer feedback-problem reports, system outages, complaints or compliments, but by then it is too late. The reliability of software must be known before it is shipped to customers. Software reliability models (SRMs) attempt to provide that information to software vendors before the software is released [4].

Software reliability growth models (SRGMs) is one of the most well-known (SRMs), it bases on failure detection during a test phase. The parameters of (SRGMs) are generally unknown and have to be estimated based on collected failure data. Two of the most popular estimation techniques are Maximum Likelihood Estimation (MLE) and Least Squares Estimation (LSE) [5], these two methods are suitable for linear problems but most of the Software Reliability Growth Models are nonlinear, so the researchers are finding many other methods for parameter estimations [6].

In recent years, the meta- heuristics algorithms have gained popularity in solving the optimization problem of scientific fields [5]. Therefore, in this work, we will use the Grey Wolf Optimizer (GWO) to estimate the parameter of the (SRGMs). Then, we will attempt to hybrid (GWO) with Real Coded Genetic Algorithm (RGA) to improve the performance of estimations. The rest of this paper is organized as follows:

Section 2 surveys various types of SRGMs and past researches that we will compare our work with. In Section 3 and 4, GWO and RGA are explained. In section 5, the proposed HGWO was introduced for parameters estimation. Then, the experimental results based on two groups of datasets are presented and discussed in Section 6. Finally, some conclusions are given in Section 7.

2. LITERATURE REVIEW

2.1 Survey of SRGMs

Through the last years, many software reliability growth models (SRGMs) have been suggested and investigated for measuring the software reliability.

(SRGMs) are mathematical models that represent software failures as a random process and can be used to evaluate development status during the test phase; most of (SRGMs) depend on some assumptions and distributions [7].

There are some classifications of (SRGMs) and the following classification according to the modeling strategy which was used in the model definition [8]: (1) General Order Statistics Models whose models are described with respect to the failure times. (2) Non-homogenous Poisson process (NHPP) whose models are described with respect to the number of observed failures.

There are many software reliability growth models, but the most frequently used is Non Homogeneous Poisson process Model (NHPP model) which shows more accuracy than the other models [9].

These models can help software practitioners to make decisions like the reliability of a software product has reached an accepted limit and when the software system is ready for release [10].

(NHPP) models are also called failure count models which are based on the number of failures that happen in different time intervals. The number of failures that detected is modeled as a stochastic process, where N(t) indicates the number of failures that have occurred at time t. What we mean by the Poisson process is non-homogenous is that the failure intensity is not constant, which means that the expected number of faults found at time t cannot be described as a function linear in time (an ordinary Poisson process can be described by that) [11]. NHPP models assume that the number of defects discovered during the time (t) follows (NHPP) with mean value function μ(t). The mean value function derivative leads to λ(t) which is the failure intensity function of the software that usually decreases as faults are detected and removed [12]. There are many (NHPP) models; we will explain four of them that were used in this work, namely:

2.1.1 Goel-Okumoto Model (G_O):

This model was first introduced by Goel and Okumoto in (1979), it also called (Exponential NHPP Model) [12, 13]. The μ(t) and λ(t) can be given as:

$$\mu(t) = a\left(1 - e^{-bt}\right) \quad \ldots\ldots (1)$$

$$\lambda(t) = abe^{-bt} \quad \ldots\ldots (2)$$

Where:

(a) Denote the initial estimate of the total failure recovered at the end of the testing process.
(b) Represents fault detection rate.
(t) Time of failure.

2.1.2 Power Model (POW):

It is one of the oldest models suggested by Duane in (1964). Basically is a graphical approach to perform analysis of reliability growth data and it is simple and easy to understand [6]. The μ(t) and λ(t) can be given as:

$$\mu(t) = at^{b} \quad \ldots\ldots (3)$$

$$\lambda(t) = abte^{b-1} \quad \ldots\ldots (4)$$

2.1.3 Delayed S-shaped Model (DSS) [12, 14]:

It was first introduced in Yamada et al. in (1983). The μ(t) and λ(t) can be given as:

$$\mu(t) = a(1 - (1 + bt)e^{-bt}) \quad \ldots\ldots (5)$$

$$\lambda(t) = ab^{2}te^{-bt} \quad \ldots\ldots (6)$$

2.1.4 Inflection S-shaped Model (INFS):

Was proposed by Ohba in (1984). The μ(t) and λ(t) can be given as:

$$\mu(t) = \frac{a\left(1-e^{-bt}\right)}{1+ce^{-bt}} \quad \ldots\ldots (7)$$

$$\lambda(t) = \frac{abe^{-bt}\,(1+c)}{\left(1+ce^{-bt}\right)^{2}} \quad \ldots\ldots (8)$$

$$c = \frac{1-r}{r} \quad \ldots\ldots (9)$$

Where:
(c) Denote inflection parameter.

(1>r>0) Called the inflection rate that indicates the ratio of detectable faults to the total number of faults in the software.

(c) Denote inflection parameter.

(1>r>0) Called the inflection rate that indicates the ratio of detectable faults to the total number of faults in the software.

The model becomes equivalent to the Exponential Growth Model if the inflection rate equals to 1, which is the same to assuming that from the beginning of a test all faults of a program are detectable. The model approaches the Logistic Curve Model as the inflection rate lean towards 0, which is the same to assuming that at the beginning of a test only a few faults of a program are detectable and faults rapidly become detectable [15].

The first three models have two parameters to be estimated (a and b) whereas the (INFS) model has three parameters to be estimated (a, b and c).

2.2 Related Work

(SRGMs) were frequently studied throughout the literatures, so here we will explain the studies that we will compare our work with them:

2.2.1. Sheta's Study [16]:

Used Particle Swarm Optimization (PSO) to solve the parameter estimation problem.

2.2.1.1 Experimental Data:

A Test/Debug dataset of 111 measurements is used. By taking 70% of the measurements to estimate the model parameters and the rest of the dataset is used to validate the developed model.

2.2.1.2 Fitness Function:

Root Mean Square Error (RMSE) is used to give each solution fitness value and whenever the (RMSE) is less that means the best solution we have.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(m_i - \mu_i)^2} \quad \ldots\ldots (10)$$

Where:
(N) Represents the number of measurements used for estimating the model parameters.

(m_i) Is the actual failure number in time t.

(μ_i) Is the expected failure number by time t.

2.2.1.3 Models:
Three models were used: G_O, POW and DSS.

2.2.1.4 The tuning parameters:
The tuning parameters and search space for PSO are given in Table 1.

Table 1: The tuning parameters for the PSO

Operator	value
Domain of search for *a*	[-1000,1000]
Domain of search for *b*	[-1,1]
Maximum Cycle Number (MCN)	1000

2.2.2 Sharma's et al. Study [17]:
Propose Dichotomous Artificial Bee Colony (DABC) to estimate the parameters of (SRGMs) and compared it with original (ABC). The study used the same data, fitness function, models and tuning parameters that used in [16].

2.2.3 Chao-Jung Hsu and Chin-Yu Huang's Study [5]:
Proposed a Modified Genetic Algorithm (MGA) to estimate the parameters of (SRGMs) and compared it with Classical Genetic Algorithm (CGA).

2.2.3.1 Experimental Data:
Three sets of real software failure data are used. The detailed information of these datasets is listed in Table 2.

Table 2: Software Failure Data

Dataset	Time Units	No. of failures
DS1 [15]	19 weeks	328
DS2 [18]	20 weeks	100
DS3 [19]	34 weeks	181

2.2.3.2 Fitness Function:
Least Squares Estimation (LSE) is used to give each chromosome a fitness value and the bigger value of fitness function means better solution. The fitness function is as follow:

$$fitness_{LSE} = \frac{1}{\sum_{i=1}^{N}(m_i - \mu_i)^2} \quad \ldots\ldots\ldots\ldots\ldots(11)$$

The Mean Square Error (MSE) is used for quantitative comparisons. It is defined as:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(m_i - \mu_i)^2 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots (12)$$

A smaller MSE indicates a smaller fitting error, and better overall performance.

2.2.3.3 Models:
Two models were used: G_O and INFS.

2.2.3.4 The tuning parameters:
The tuning parameters and operators for (CGA and MGA) are given in Table 3.

Table 3: The tuning parameters for the MGA and CGA

	MGA	CGA
Chromosome Representation	Binary Encoding 32-Bit	Binary Encoding 16-Bit
Selection	Roulette Wheel Selection + Rebuilding	Roulette Wheel Selection
Crossover	Uniform Crossover	
Mutation	Weighted Bit Mutation	Bit Mutation
Maximum Generations No.	1000	
Population Size	100	
Crossover Rate	0.5	
Mutation Rate	0.1	
No. Of Runs	100	

3. GREY WOLF OPTIMIZER (GWO)
A new swarm intelligence algorithm introduced by Mirjalili in (2014) inspired by grey wolves (Canis lupus). This algorithm simulates the leadership grading and hunting mechanism of grey wolves in nature. Four types of grey wolves (alpha α, beta β, delta δ , and omega ω) are used to simulating the leadership grading [20]. The leader is called alpha, the alpha is frequently responsible for making the essential decisions about sleeping place, hunting, time to wake, and so on. The second level in the grading of grey wolves is beta. The betas are secondary wolves that help the alpha in decision-making or other activities. The lowest ordering grey wolf is omega. Omega wolves always have to bow to all the other overriding wolves. If a wolf is not an alpha, beta, or omega, it is called delta. Delta wolves have to submit to alphas and betas, but they control the omega.

The grey wolf hunting has three main steps as follows [21]:

1) Tracking, chasing, and approaching the prey.
2) Pursuing, encircling, and harassing the prey until it stops moving.
3) Attack towards the prey.

3.1 Mathematical Model for the Algorithm
3.1.1 Search for prey (exploration):
Grey wolves frequently search according to the position of the leader alpha (first best solution), beta (second best solution), and delta (third best solution). They separate from each other to search for prey and converge to attack the prey [20].

3.1.2 Encircling prey [22]:
The following equations are offered to mathematically model encircling behavior of grey wolves:

$$\vec{D} = \left|\vec{C} \, . \, \vec{X}_p\,(t) - \vec{X}\,(t)\right| \quad \ldots\ldots\ldots\ldots\ldots\ldots(13)$$

$$\vec{X}\,(t+1) = \vec{X}_p\,(t) - \vec{A} \, . \vec{D} \ldots\ldots\ldots\ldots\ldots\ (14)$$

Where:

(t) Indicates the current cycle.
(A) and (C) Are coefficient vectors.
(X_p) Is the position vector of the prey.
(X) Is the position vector of a grey wolf.
The vectors A and C are calculated as follows:

$$\vec{A} = 2\vec{a} \, . \vec{r_1} - \vec{a} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(15)$$

$$\vec{C} = 2 \, . \vec{r_2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(16)$$

Where:
(a) Is linearly decreased from 2 to 0 over the course of cycles.
(r1 and r2) are random vectors in [0, 1].

3.1.3 Hunting [20]:
There is no clue around the location of the solution (the prey in this case) in the search space, so to mathematically mimic (first best candidate solution) beta (second best candidate solution), and delta (third best candidate solution) have better

knowledge about the potential location of prey. Therefore, these three best solutions are saved and the other search agents update their positions according to the position of the best search agents. The following equations are for position update:

$$\vec{D_\alpha} = |\vec{C_1} . \vec{X_\alpha} - \vec{X}| \quad \text{....(17)}$$

$$\vec{X_1} = \vec{X_\alpha} - \vec{A_1} . (\vec{D_\alpha}) \quad \text{....(18)}$$

$$\vec{D_\beta} = |\vec{C_2} . \vec{X_\beta} - \vec{X}| \quad \text{....(19)}$$

$$\vec{X_2} = \vec{X_\beta} - \vec{A_2} . (\vec{D_\beta}) \quad \text{....(20)}$$

$$\vec{D_\delta} = |\vec{C_3} . \vec{X_\delta} - \vec{X}| \quad \text{....(21)}$$

$$\vec{X_3} = \vec{X_\delta} - \vec{A_3} . (\vec{D_\delta}) \quad \text{....(22)}$$

$$\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3} \quad \text{....(23)}$$

3.1.4 Attacking prey (exploitation)[22]:
The grey wolves finish the hunt by attacking the prey when it stops moving. To mathematically model attacking the prey, the value of (a) is linearly decreased from 2 to 0 over the course of cycles.

$$\vec{a} = 2 - cycle * (2/MCN) \quad \text{....(24)}$$

The general steps of the GWO algorithm are illustrated in Figure 1 [20]:

```
Grey Wolf Optimizer
Begin
  Initialize the grey wolf population $X_i$ (i=1,2,….., n)
  Initialize a, A, and C
  Calculate the fitness of each search agent
  $X_\alpha$= the best search agent
  $X_\beta$= the second best search agent
  $X_\delta$= the third best search agent
  While (t < Max number of iteration)
    For each search agent
      Update the position of current search agent
    End for
    Update a, A, and C
    Calculate the fitness of each search agent
    Update $X_\alpha$, $X_\beta$ and $X_\delta$
    t=t+1
  End while
  Return $X_\alpha$
End
```

Fig. 1: Pseudo code of the (GWO)

4. REAL CODED GENETIC ALGORITHM (RGA)
(GA) is a searching technique, firstly studied by John Holland in the early (1970). Apparently, GA is based on the mechanism of natural evolution '*survival of the fittest*', this concept can be useful to find the solution of complicated optimization problems [23].

4.1 Chromosome Representation:
There are many types of encoding and we will explain the type that we used in this work [24]:

4.1.1 Value Encoding
Each chromosome is represented as the string of specific value. This Value can be float number, integer, character or some object. In this work, we used encoding with real numbers.

4.2 Genetic Algorithm Cycle:
As shown in Figure 2, (GA) has the following steps [25]:

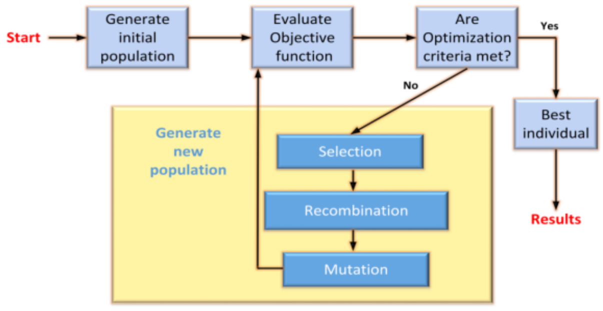

Fig. 2: (GA) cycle

4.2.1 Generate Initial Population:
The initial populations are generated randomly. Each individual in the population is called a chromosome which represents a possible solution for the problem to be solved.

4.2.2 Evaluation:
Evaluation is made by defining fitness function for each chromosome, this fitness function is an indicator that shows how close this chromosome is to the desired solution [5].

4.2.3 Termination method:
The Termination method determines when the genetic process will stop evolving. In this work, the genetic process will end either if there is no change in the population best fitness for 10 generations, or maximum number of generations has been reached [25].

4.2.4 Generate New Population:
Involves the following three steps:

1) Selection:
Selection is used to choose the fittest chromosomes from the population, these chosen chromosomes will create offsprings for the next generation [26].

There are many types of selection and we will explain the type that we used in this work.

➢ Top- Mate Selection
The first parent is selected by the fitness order, whereas the second parent is selected randomly [25].

2) Crossover:
Crossover is a process of exchange some genes between two selected parents chromosomes to create two new chromosomes (offsprings) [27]. There are many types of crossover and we will explain the type that we used in this work:

➢ Heuristic Crossover
This type of crossover is used with chromosomes with value encoded. Two offsprings chromosomes will be generate from two parents chromosomes. The first offspring is the parent whose fitness is better than the other parent, this parent is passed over to the next generation without any processing, so,

the generated offspring is a copy of its parent. The second offspring is generated from manipulation of the other parent as shown in the following equations [28, 29]:

$fitness_{parent1}$ *is better than* $fitness_{parent2}$

$$offspring_1 = parent_1 \quad \text{………..…} \quad (25)$$

$$offspring_2 = parent_1 \pm r * (parent_1 - parent_2) \quad \text{…..…. } (26)$$

Where (r) is a random value between 0 and 1.

3) Mutation:
Mutation is a process of altering gene(s) in offspring chromosome [27]. There are many types of mutation and we will explain the type that we used in this work:

➢ Non-Uniform Mutation
The value of the parent chromosome is altered in a limited range by considering the number of the current generation. The relationship between the range of changing the chromosome and the number of the current generation is opposite, when the number of the current generation is small, the range of changing the chromosome is large. As the generations pass, the range decreases [30]. The equation is as follow:

$$offspring = \begin{Bmatrix} parent_i + (upperbound_i - parent_i) * f(G) \\ or \\ parent_i - (parent_i - lowerbound_i) * f(G) \end{Bmatrix} \text{…..}(27)$$

Where:

f(G): is the range function considering the number of the current generation (G). The function *f*(*G*) is as follows:

$$f(G) = \left(r * (1 - \frac{G}{G_{max}})\right)^b \quad \text{……..…………..}(28)$$

Where:
(G_{max}): Is the maximum number of generations
(*b*): Is a shape parameter.
(*r*): Is a uniform random number between 0 and 1.

The general steps of the RGA are illustrated in Figure 3:

```
Real Coded Genetic Algorithm
Begin
Generate the initial population of chromosomes
Define fitness function f (x), x = (x_1, x_2, ..., x_d)
Calculate fitness function of all individual
chromosomes
Select parents by top-mate selection
Initial probabilities of crossover (pc) and
mutation (pm)
While (t < Max Generation) or (stop criterion);
    If pc >rand
        Generate new solution by Heuristic
        Crossover
    End if
    If pm >rand
        Generate new solution by Non-Uniform
        Mutation
    End if
    Put the solutions in the new generation
End while
End
```

Fig. 3: Pseudo code of the (RGA)

5. PROPOSED HYBRID GWO WITH RGA (HGWO)

The inspiration of developing (HGWO) approach is to chain the advantages of both (GWO) algorithm and (RGA) to obtain a hybrid algorithm that is easy to implement and has a good balance between global search and local search and has a fast convergence. In the proposed (HGWO), the total numbers of iterations are equally shared by both (RGA) and (GWO). In the first step, the first half of the iterations are given to (RGA) that explores the global search place, then the solution that obtained from (RGA) is given to (GWO). In the second step (GWO) explores search space starting with the solution obtained by RGA that is set as initial population of (GWO) and continue the manipulation to find new enhanced solutions. Figure 4 represents the flowchart of the hybrid algorithm.

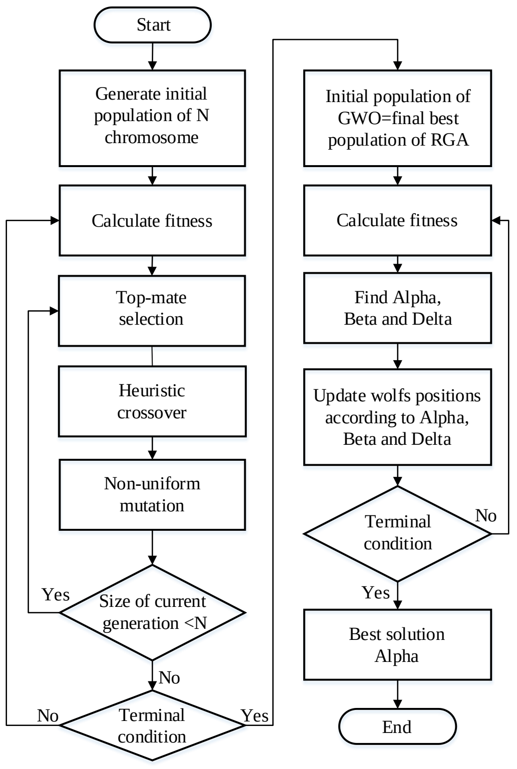
Start
Generate initial population of N chromosome
Calculate fitness
Top-mate selection
Heuristic crossover
Non-uniform mutation
Size of current generation <N
Yes
No
Terminal condition
No
Yes
Initial population of GWO=final best population of RGA
Calculate fitness
Find Alpha, Beta and Delta
Update wolfs positions according to Alpha, Beta and Delta
Terminal condition
No
Yes
Best solution Alpha
End

Fig. 4: The flowchart of the (HGWO)

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1 Experimental Dataset:

In this work, the parameters of SRGMs are estimated by using two groups of datasets accordance to those referenced by other researchers with which the comparisons were made.

6.2 Comparison with Other Algorithms

6.2.1 By using the first group of dataset

Two comparisons were made to test the efficiency of the search algorithm employed in this work:

1) A comparison between (GWO) and the algorithms in ([16] and [17]) are made.

2) Then, (HGWO) is compared with the (GWO) to see the improvement of the original algorithm.

The tuning parameters for the (GWO) for the first comparison are in Table 4.

Table 4: The tuning parameters for the GWO

Operator	Value
Domain of search for *a*	[-1000,1000]
Domain of search for *b*	[-1,1]
Domain of search for *c*	[0,1000]
Search dimensions	3 for INFS 2 for rest models
No. of search agents	20
Maximum Cycle Number	1000

Three models were used: (G_O, POW and DSS). For the comparison criteria, (RMSE) is used. Here (GWO) is run for one time.

Results in Table 5 show the (RMSE) for (PSO, ABC, DABC and GWO), as we mentioned before whenever the (RMSE) is less that means the best solution we have. The (GWO) outperformed the (PSO) and (ABC) for all models and outperformed (DABC) in (DSS) model only.

For the second comparison between (HGWO) and (GWO) the maximum number of iterations for (HGWO) is set to 1000, and as we mentioned before this number is equally shared by (RGA) and (GWO) (500 iteration for each). First, the (RGA) is executed and the result will set as initial solution for (GWO). Tuning parameters for the (RGA) are in Table 6.

Table 6: The tuning parameters for the RGA

Operator	Value
Domain of search for *a*	[-1000,1000]
Domain of search for *b*	[-1,1]
Domain of search for *c*	[0,1000]
No. of chromosomes	20
Maximum No. of generations	500
Chromosome representation	Value encoding
Selection	Top-mate selection
Crossover	Heuristic Crossover
Mutation	Non-Uniform Mutation
Crossover rate	0.5
Mutation rate	0.1

For (GWO), the tuning parameters are the same as in Table 4 except that the maximum cycle number is set to 500. The total number of iterations needed by (HGWO) to reach optimal solution is the sum of iterations needed by (RGA) and (GWO).

Results in Table 7 show the (RMSE) for (GWO and HGWO) by using three models: (G_O, POW and DSS). The origin and hybrid algorithm reach nearly the same (RMSE) but (HGWO) needs less iterations to reach it.

Table 5: Comparison GWO with (PSO, ABC and DABC) using first group of dataset

Model	**RMSE-testing (30%of data)**				**Best values by GWO**	
	PSO	**ABC**	**DABC**	**GWO**	**a**	**b**
G_O	80.896	119.642	72.018	77.901	684.424	0.017
POW	149.96	158.675	81.923	146.33	22.388	0.728
DSS	17.063	17.091	29.805	16.667	501.897	0.063

Table7: Comparison GWO with HGWO using first group of dataset

Model	**GWO**		**Hybrid GWO**			
	RMSE-testing	**No. Of cycles**	**RMSE-testing**	**No. Of cycles by RGA**	**No. Of cycles by GWO**	**Total no. of cycles by HGWO**
G_O	77.901	978	77.893	30	486	516
POW	146.330	961	146.426	30	475	505
DSS	16.667	987	16.497	20	493	513

Figure 5 illustrate the convergence fitness function for (GWO) and (HGWO) for the three models. In G_O model both algorithms converge at fitness value 25.1845 but (HGWO) needs less iteration than the needed by (GWO) to reach this fitness value. Also for POW model and DSS model the (HGWO) converge faster than (GWO) at fitness values equal to 32.9521 and 20.7244 respectively.

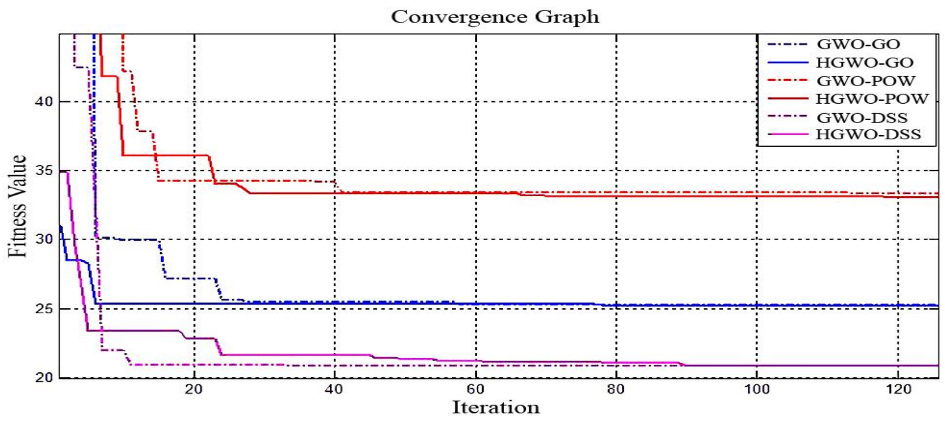

Fig. 5: convergence graph for GWO and HGWO for first group of dataset

6.2.2 By using the second group of dataset

Two comparisons were made to test the efficiency of the search algorithm employed in this work:

1) A comparison between (GWO) and the algorithms in ([5]) are made.

2) Then, the (HGWO) is compared with the (GWO) to see the improvement of the original algorithm.

For the first comparison, two models were used: (G_O and INFS). For the comparison criteria, (MSE) is used. (GWO) is run 100 times repeatedly, and the minimum (MSE) of each model in all dataset (DS1, DS2 and DS3) is obtained. Then, the average values for reaching the minimum value of (MSE) are collected. Notice that the number of generations is rounded to an integer.

In Tables 8, 9 and 10 results show the (MSE) for (CGA, MGA and GWO), all three algorithms reach the same (MSE) but (GWO) outperformed the (CGA) and (MGA) for all models according to the average number of generations needed to reach optimal solution.

Table 8: Comparison GWO with (CGA and MGA) using DS1

Data set	Model	CGA		MGA		GWO		Best values found by GWO		
		MSE	Av. No. Of gen.	MSE	Av. No. Of gen.	MSE	Av. No. Of gen.	a	b	c
DS1	G_O	139.815	13415	139.815	5341	139.8151	834	760.5316	0.0323	-
	INFS	82.704	78664	82.704	15664	82.7040	891	382.3867	0.1788	2.886

Table 9: Comparison GWO with (CGA and MGA) using DS2

Dataset	Model	CGA		MGA		GWO		Best values found by GWO		
		MSE	Av. No. Of gen.	MSE	Av. No. Of gen.	MSE	Av. No. Of gen.	a	b	c
DS2	G_O	11.617	48104	11.617	7331	11.6171	42	130.2074	0.0832	-
	INFS	8.98	77889	8.98	17490	8.9792	268	110.8352	0.1720	1.2047

Table 10: Comparison GWO with (CGA and MGA) using DS3

Dataset	Model	CGA		MGA		GWO		Best values found by GWO		
		MSE	Av. No. Of gen.	MSE	Av. No. Of gen.	MSE	Av. No. Of gen.	a	b	c
DS3	G_O	22.863	39209	22.863	3350	20.3895	38	1000	0.0061	**-**
	INFS	5.82	83578	5.82	20749	5.8200	684	229.3826	0.0875	3.6893

For the second comparison, (HGWO) is run 100 times repeatedly, for each run (MSE) is obtained and the needed iterations by (RGA) and (GWO) is added to obtain the total iterations needed by (HGWO) to reach this (MSE). After we gained 100 (MSE), the minimum (MSE) of each model in all dataset (DS1, DS2 and DS3) is obtained. Then, the average values for reaching the minimum value of (MSE) are collected. Notice that the number of generations is rounded to an integer.

Results in Table 11 show that (HGWO) reach nearly the same (MSE) for (GWO) but in less iteration.

Table 11: Comparison GWO with (CGA and MGA) using second group of dataset (DS1, DS2 and DS3)

Data set	Model	GWO		HGWO			
		MSE-testing	Av. No. Of gen.	MSE-testing	No. Of cycles by RGA	No. Of cycles by GWO	Av. No. Of gen. by HGWO
DS1	G-O	139.8151	834	139.8151	50	458	367
	INFS	82.7040	891	82.7467	60	494	548
DS2	G-O	11.6171	42	11.6171	20	2	30
	INFS	8.9792	268	8.9798	20	95	82
DS3	G-O	20.3895	38	20.3895	20	9	28
	INFS	5.8200	684	5.8210	20	64	84

Figure 6 illustrate the convergence fitness function for (GWO) and (HGWO) for the two models using DS1. Both algorithms converge at fitness value equal to 3.76×10^{-4} for G_O model and 6.36×10^{-4} for INFS model but (HGWO) needs less iteration than the needed by (GWO) to reach these fitness values.

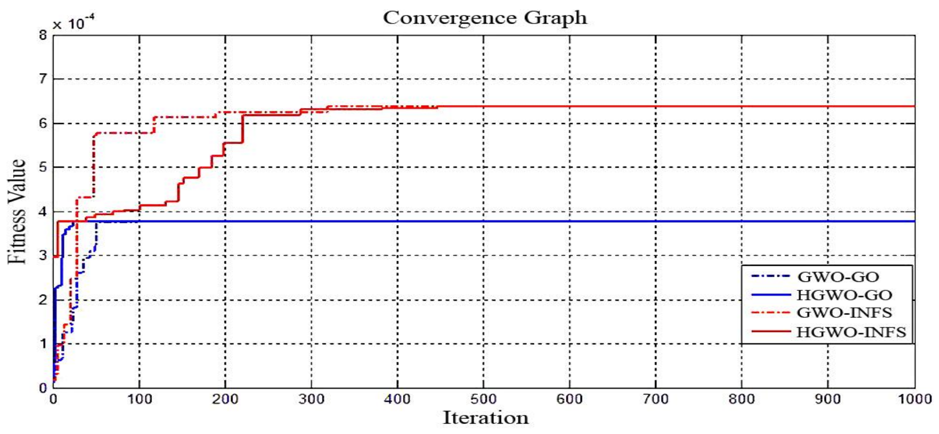

Fig. 6: convergence graph for GWO and HGWO for DS1

Figure 7 illustrate the convergence fitness function for (GWO) and (HGWO) for the two models using DS2. Both algorithms converge at fitness value equal to 4.3×10^{-3} for G_O model and 5.6×10^{-3} for INFS model but (HGWO) needs less iteration than the needed by (GWO) to reach these fitness values.

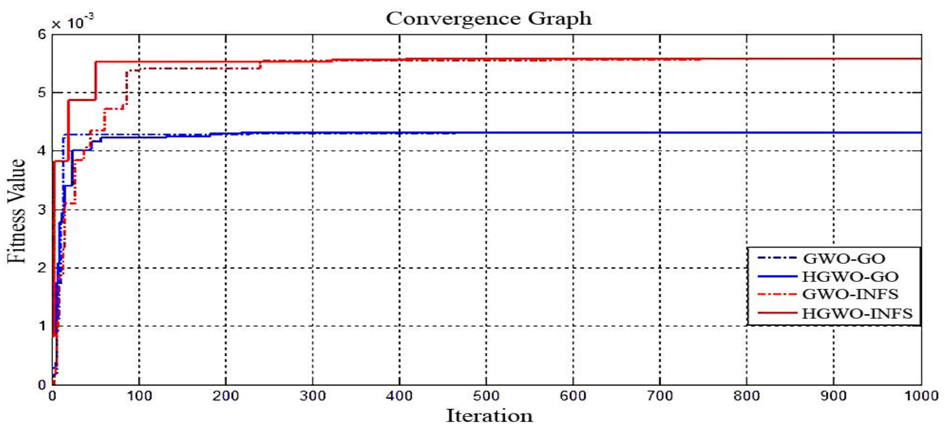

Fig. 7: Convergence graph for GWO and HGWO for DS2.

In Figure 8, both (GWO) and (HGWO) converge at the same value (1.4×10^{-3} for G_O model, 5.1×10^{-3} for INFS model) but the (HGWO) converge faster than (GWO).

Figure 8 illustrate the convergence fitness function for (GWO) and (HGWO) for the two models using DS3. Both algorithms converge at fitness value equal to 1.4×10^{-3} for G_O model an 5.1×10^{-3} for INFS model but (HGWO) needs less iteration than the needed by (GWO) to reach these fitness values.

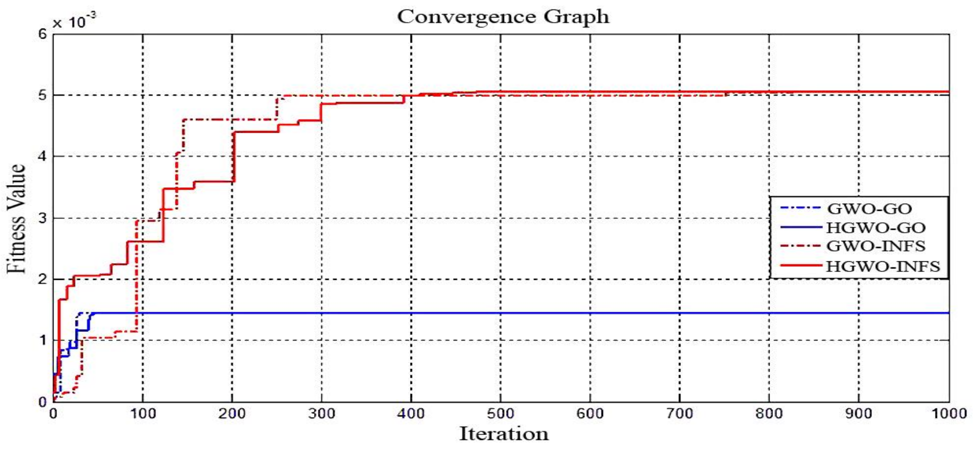

Fig. 8: Convergence graph for GWO and HGWO for DS3

7. CONCLUSION AND FUTURE WORK

In this work, (GWO) and (HGWO) were used to estimate the parameters of four (SRGMs) models: G_O, POW, DSS and INFS by using two groups of datasets. Two comparisons were made for each group of datasets, the first comparison compare (GWO) with other algorithms: PSO, ABC, DABC, CGA and MGA and the results have shown that the (GWO) present more accurate solution when it's compared with these algorithms. The second comparison compares (GWO) with the proposed (HGWO) which is a hybridization between genetic algorithm and original (GWO). The comparisons between the origin and the hybrid algorithm show that both algorithms present optimal solutions but (HGWO) needs less iteration than (GWO) to reach the solution, so the enhancement gained from the hybridization process lay in speeding the process of finding the best solution.

As for further recommendations, many other swarm intelligence can be used for parameters estimation problem and compared to our work. Future work might also include a different hybrid method with (RGA) or hybrid with other swarm algorithms for better performance.

8. REFERENCES

[1] Sheakh, T. H., Singh, V. P., 2012,"Taxonomical Study Of Software Reliability Growth Models", International Journal of Scientific and Research Publications, Volume 2, Issue 5, pp.1-3.

[2] Kaswan, K.S., Choudhary , S., Sharma, K., 2015," Software Reliability Modeling using Soft Computing Techniques: Critical review", J Inform Tech Softw Eng 5: 144.

[3] Xie, M., Dai, Y. S., Poh, K. L., 2004, "Computing System Reliability Models and Analysis", Springer, ISBN-10: 030648496X, ISBN-13:978-0306484964, pp.1-293.

[4] Wood, A., 1996, "Software Reliability Growth Models", Tandem Tech., Technical Report, Vol. 96.1, Tandem Computers Inc., Corporate Information Center, Cupertino Calif., Part Number 130056.

[5] Hsu, C.J., Huang, C.Y., 2010," A Study on the Applicability of Modified Genetic Algorithms for the Parameter Estimation of Software Reliability Modeling " , IEEE 34th Annual Computer Software and Applications Conference, pp.531-540.

[6] Shanmugam, L., Florence, L., 2012, "A Comparison of Parameter Best Estimation Method for Software Reliability Models", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.5, pp.91-102.

[7] Su, Y.S., Huang, C.Y., 2006," Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models ", The Journal of Systems and Software 80, pp.606–615.

[8] AL-Saati, N., Abd-AlKareem, M.,2013," The Use of Cuckoo Search in Estimating the Parameters of Software Reliability Growth Models", International Journal of Computer Science and Information Security, Vol. 11, No. 6.

[9] Kelanibandara, K.W.K.B.P.L.M.,2012," Software Reliability Estimation Using Cubic Splines Network Model", thesis, University of Colombo School of Computing , pp.1-72.

[10] Lai, R., Garg, M., 2012, "A Detailed Study of NHPP Software Reliability Models", Journal of Software, Vol.7, No.6, pp.1296-1306.

[11] Wohlin, C., Höst, M., Runeson, P., Wesslén, A., 2001, " Software Reliability", Encyclopedia of Physical Science and Technology, Volume 15, pp.1-27.

[12] Meyfroyt, P. H. A., 2012,"Parameter Estimation for Software Reliability Models", thesis, Eindhoven: Technische Universiteit Eindhoven, pp.1-65.

[13] Song, K. Y., Chang, I. H., 2014," Parameter Estimation and Prediction for NHPP Software Reliability Model and Time Series Regression in Software Failure Data", J. Chosun Natural Sci., Vol. 7, No. 1, pp. 67 – 73.

[14] Williams, P., 2006,"prediction capability analysis of two and three parameters software reliability growth models", information technology journal 5(6), pp.1048-1052.

[15] Ohba, M., 1984,"software reliability analysis models", IBM J. RES. DEVELOP. VOL. 28 NO. 4, pp.228-443.

[16] Sheta, A. F., 2007, " Parameter Estimation of Software Reliability Growth Models by Particle Swarm Optimization", AIML Journal, Volume (7), Issue (1), pp.55-61.

[17] Sharma, T.K., Pant, M., Abraham, A., 2011," Dichotomous Search in ABC and its Application in Parameter Estimation of Software Reliability Growth Models ", Third World Congress on Nature and Biologically Inspired Computing, pp.214-219.

[18] Wood A., 1996,"Predicting Software Reliability," IEEE Computer, vol. 29, no. 11, pp. 69-77.

[19] Jeske, D. R., Zhang, X., Pham, L., 2005," Adjusting Software Failure Rates That Are Estimated From Test Data ", IEEE TRANSACTIONS ON RELIABILITY, VOL. 54, NO. 1, pp.107–114.

[20] Mirjalili, S. A., Mirjalili , S. M., Lewis, A., 2014," Grey Wolf Optimizer", Advances in Engineering Software 69, pp. 46–61.

[21] Madadi, A., Motlagh, M. M., 2014," Optimal Control of DC motor using Grey Wolf Optimizer Algorithm", Tech J Engin & App Sci., 4 (4): 373-379.

[22] Mirjalili, S. A., 2015," How effective is the Grey Wolf optimizer in training multi-layer perceptrons",The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies 43:645.

[23] HERRERA, F., LOZANO, M., VERDEGAY, J.L., 1998," Tackling Real Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis", Artificial Intelligence Review 12: 265–319.

[24] Kumar, A., 2013," ENCODING SCHEMES IN GENETIC ALGORITHM", International Journal of Advanced Research in IT and Engineering, Vol.2 ,No.3, pp. 1-7.

[25] AL Neamy, J. S., 2006,"Brain Tumors Images Diagnosis Using Hybrid Intelligency Techniques", Ph.D. Thesis, college of computers and mathematics science/university of Mosul.

[26] Goldberg, D. E., Deb, K., 1991, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms", pp.70-92.

[27] Achiche, S., Baron, L., Balazinski, M., 2004," Real/binary-like coded versus binary coded genetic algorithms to automatically generate fuzzy knowledge bases: a comparative study", S. Achiche et al. / Engineering Applications of Artificial Intelligence 17, pp.313–325.

[28] Peltokangas, R., Sorsa, A., 2008," Real-coded genetic algorithms and nonlinear parameter identification", Report A No 34, pp.1-28.

[29] KAYA, Y., UYAR, M., TEKDN, R., 2011, "A Novel Crossover Operator for Genetic Algorithms: Ring Crossover".

[30] Michalewicz,Z.,1996,"Genetic Algorithms Data Structures =Evolution Programs", springer-verlag Berlin Heidelberg New York.