Estimating the Parameters of Software Reliability Growth Models Using the Grey Wolf Optimization Algorithm

Alaa F. Sheta[†] and Amal Abdel-Raouf[†‡]
[†]Computers and Systems Department, Electronics Research Institute, Giza, Egypt
[‡]Computer Science Department, Southern Connecticut State University, USA

***Abstract*—In this age of technology, building quality software is essential to competing in the business market. One of the major principles required for any quality and business software product for value fulfillment is reliability. Estimating software reliability early during the software development life cycle saves time and money as it prevents spending larger sums fixing a defective software product after deployment. The Software Reliability Growth Model (SRGM) can be used to predict the number of failures that may be encountered during the software testing process. In this paper we explore the advantages of the Grey Wolf Optimization (GWO) algorithm in estimating the SRGM's parameters with the objective of minimizing the difference between the estimated and the actual number of failures of the software system. We evaluated three different software reliability growth models: the Exponential Model (EXPM), the Power Model (POWM) and the Delayed S-Shaped Model (DSSM). In addition, we used three different datasets to conduct an experimental study in order to show the effectiveness of our approach.**

***Index Terms*—Software Reliability, Reliability Growth Models, Grey Wolf Optimizer, Exponential Model, Power Model, Delayed S-Shaped Model**

I. Introduction

With the increasing importance of software systems in almost all aspects of our lives, there is a great need for the production of high quality software systems. The traditional model of software quality factors, suggested by McCall [1], consists of eleven different factors that should be considered in determining the quality of software. Subsequent models include Evans and Marciniak [2], which consists of twelve factors, and Deutsch and Willis [3], which consists of fifteen factors. All of these models incorporate reliability as one of the software quality factors. Software reliability is defined according to [4] as: "the probability, over a given period of time, that the system will correctly deliver services as expected by the user." A more precise definition of software reliability is given by [5], [6] as: "the probability of failure-free operation over a specified time, in a given environment, for a specific purpose." Unfortunately, the task of identifying and repairing software faults is costly. Moreover, the cost of finding the remaining faults increases as the number of faults decreases until the cost exceeds the benefit [7], [8]. Therefore, there is no software system that is failure-free, which is why the reliability requirements should be included in any software development contract. Software reliability is measured based on the maximum allowable rate of failure and can represent an entire system or one or more of its parts [9]–[11].

The cost of software development is always higher for more reliable systems. Consequently, the desired reliability should be determined depending on the criticalness of failure-free operation of the system. For example, the failure rate of a life-threatening system such as heart-monitor should be very low while a company website may have a higher failure rate.

In the literature, many methods are introduced to estimate and predict software reliability [12]–[20]. The proposed methods can be classified into two main categories [9]. The first category is Software Reliability Prediction Models and the second category is Software Reliability Growth Models (SRGM). Software reliability prediction models consider predicting the reliability early in the development life cycle. In the requirements, design or implementation phases, the model uses historical data and some quantitative measurements like Lines of code (LOC) and depth of nesting loops to estimate the failure rate. Examples of software reliability prediction models include the orthogonal defect classification model [21] and the constructive quality model [22], [23]. Some reliability models may be based on software architecture and others on modified adaptive testing [24], [25]. The second category, SRGM, represents how the system reliability changes over time during the testing phase and based on test data. SRGMs collect defect data and statistically correlate this data with known mathematical functions to predict software reliability [26]–[29].

Many SRGMs are proposed to represent the relationship between software reliability and time. SRGMs can be classified as either parametric or non-parametric models. The most famous parametric models are the Non-Homogeneous Poisson Process (NHPP) models used in [30]–[32]. Non-parametric models have less restricted assumptions as they can predict reliability based only on defect data [33]. Other SRGMs are introduced using Neural Networks in [14], [34], [35], using Bayesian learning in [36], [37] and using particle swarm optimization in [38], [39].

In this paper, we utilize the Grey Wolf Optimization (GWO) algorithm to predict faults during the software testing process using software faults historical data. The rest of this paper is organized as follows: In Section II, we briefly introduce

some SRGM models that we use in our study. Section III provides an overview of the GWO algorithm. Section IV shows the evaluation criterion adopted in this study. The experimental results developed for parameter estimation of software reliability are given in Section V. Finally, we provide the conclusions and future work in Section VI.

II. Reliability Growth Models (SRGM)

The inability to meet software requirements and/or deviation from the goal for which the software was developed is defined as software failure. Software reliability depends mainly on the way we handle failure. For example, detecting failure during execution and repairing it increases the reliability of the software as a function of time. This is what happens during the software testing process and before release of software to the market. Software reliability growth models (SRGMs) are the models concerned with the explanation and the description of software failures.

In the literature, many SRGMs were presented to estimate the reliability of software systems [40], [41]. Each SRGM assumes a function called $M(t)$ that measures the number of failures experienced at a given time t. The SRGM parameters are estimated based on either the failure times $t_1, t_2, \ldots$ or the times between failures $\Delta t_1, \Delta t_2, \ldots$. For a given software project, $\mu(t)$ represents the mean value function of a SRGM reflecting the expected number of failures experienced at time t. The derivative of the mean value function with respect to time, $\frac{d\mu(t)}{dt}$, is defined as the failure intensity $\lambda(t)$. In the following subsections, we briefly describe three well-known SRGM models that we use in our study.

A. Exponential Model (EXPM)

The exponential model was first provided in [5], [42]. This model is also known as the Goel-Okumoto exponential model [43] shown in Equations 1.

$$\begin{aligned} \mu(t;b) &= b_0(1-e^{-b_1 t}) \\ \lambda(t;b) &= b_0 b_1 e^{-b_1 t} \end{aligned} \tag{1}$$

The parameter b_0 is the expected total number of failures recovered at the end of the testing process (i.e. v_0). b_1 represents the rate at which the defect rate decreases (see Equation 2).

$$b_1 = \frac{\lambda_0}{v_0} \tag{2}$$

where the parameter λ_0 is the initial failure intensity and v_0 is the total failure at the end of the testing process.

B. Power Model (POWM)

The power model is also known as the Non-Homogeneous Poisson Process (NHPP) [44]. The equations that govern μ and λ are given in Equations 3.

$$\begin{aligned} \mu(t;b) &= b_0 t^{b_1} \\ \lambda(t;b) &= b_0 b_1 t e^{b_1 - 1} \end{aligned} \tag{3}$$

Many systems have adopted the NHPP model for analysis. For example in [45], author uses the NHPP to estimate software reliability for nuclear safety software. The Bayesian statistical inference (BSI) method was used to estimate the model parameters.

C. Delayed S-Shaped Model (DSSM)

This model is known as Yamada delayed S-shaped model [46], [47]. The model is a finite failure model. Yamada et al. [27] provided this model for error detection, in which the observed growth curve of the cumulative errors has an S-shape. The system equations for $\mu(t;b)$ and $\lambda(t;b)$ are given in Equation 4.

$$\begin{aligned} \mu(t;b) &= b_0(1-(1+b_1 t)e^{-b_1 t}) \\ \lambda(t;b) &= b_0 b_1^2 t^{-b_1 t} \end{aligned} \tag{4}$$

where b_0 is the expected total number of failures and b_1 represents the failure detection rate.

III. Grey Wolf Search Algorithm

The Grey Wolf Optimizer (GWO) is a meta-heuristics algorithm introduced by Mirjalili et al. [48]. The GWO is utilized to solve many optimization problems in different fields and successfully provides highly competitive results [49]–[52].

The GWO algorithm is based on the wild behavior of the grey wolves during hunting. According to the dominant hierarchy leadership order, the GWO divides the animals' population into four categories: alpha (α), beta (β), delta (δ), and omega (ω). Consequently, the optimization process, the same as the hunting, is guided by the highest rank leaders: α, β and δ respectively which represent the best three solutions in the search space. The ω wolves, the lowest in the hierarchical rank, represent the rest of the solutions that must adjust their positions to follow the other dominant wolves.

It is assumed that each candidate solution with dimension n is represented by the vector $\vec{X}$ such that the Grey wolf position vector is given as:

$$\vec{X} = \{x_1, x_2, \ldots, x_n\} \tag{5}$$

During the hunting process, the grey wolves surround the prey (i.e. solution of the problem). This surrounding behavior in GWO can be represented mathematically as follows:

$$\vec{D} = |\vec{C}.\vec{X}(t)_p - \vec{X}(t)| \tag{6}$$

$$\vec{X}(t+1) = \vec{X}_p(t) + \vec{A}.\vec{D} \tag{7}$$

where $\vec{X}(t)_p$ is the position vector of the prey, $\vec{X}(t)$ is the position vector of the Grey wolf, t is the current iteration, $\vec{A}$ and $\vec{C}$ are coefficient vectors that vary to allow the wolves to adjust their positions in the space around the prey. The coefficient vectors $\vec{A}$ and $\vec{C}$ are computed according to Equations 8.

$$\begin{aligned} \vec{A} &= 2\,\vec{a}\,\vec{r}_1 - \vec{a} \\ \vec{C} &= 2\,\vec{r}_2 \end{aligned} \tag{8}$$

It is given that the elements $\vec{a}$ are linearly decreasing from the value of 2 to the value of 0 over the search process and $\vec{r}_1, \vec{r}_2$ are random vectors selected in the domain of [0,1].

Then the GWO saves the best three solutions (alpha, beta and delta wolves) and allows the other solutions (omega wolves) to adjust their positions according to the positions of the best solutions. The following equations are used to calculate the distance between the current position and α, β, and δ, respectively (see Equations 9):

$$\begin{aligned} \vec{D}_\alpha &= |\vec{C}_1.\vec{X}_\alpha - \vec{X}| \\ \vec{D}_\beta &= |\vec{C}_2.\vec{X}_\beta - \vec{X}| \\ \vec{D}_\delta &= |\vec{C}_3.\vec{X}_\delta - \vec{X}| \end{aligned} \tag{9}$$

where $\vec{X}_\alpha, \vec{X}_\beta$ and $\vec{X}_\delta$ are the positions of the α, β and δ, respectively, $\vec{X}$ is the position of the current solution and $\vec{C}_1, \vec{C}_2$ and $\vec{C}_3$ are random vectors. Then, the final position of the current solution can be calculated as in Equation 10.

$$\begin{aligned} \vec{X}_1 &= \vec{X}_\alpha + \vec{A}_1.\vec{D}_\alpha \\ \vec{X}_2 &= \vec{X}_\beta + \vec{A}_2.\vec{D}_\beta \\ \vec{X}_3 &= \vec{X}_\delta + \vec{A}_3.\vec{D}_\delta \end{aligned} \tag{10}$$

Thus, $\vec{X}(t+1)$ can be computed as follows:

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{11}$$

where t represents the number of iterations and $\vec{A}_1, \vec{A}_2$ and $\vec{A}_3$ are random vectors that vary to allow the wolves to attack towards the prey. Finally, the hunting process ends when the grey wolves attack the prey after it stops moving. In the next sections we show how to utilize the GWO to estimate the parameters for number of SRGMs.

IV. Measure for Model Predictability

To make a comparison between different SRGMs it is important to measure the model accuracy in terms of some meaningful measurements. In our case we adopt the Goodness-of-fit criteria. These criteria are applied to measure the quality of the solution provided and determine the proximity of the estimated failures to the measured failures.

Assume we have N measurements which represent the cumulative number of failures found at time t_i where t_i is the accumulated execution time. Then $\mu(t_i, b)$ can be defined as the projected number of failure at time t_i by a model.

According to the Goodness-of-fit criterion, a curve corresponding to a selected model is fitted to all data points $t_i, \mu_i,\ i = 1, \ldots, n$; then the difference between the actual measured failures y and the estimated failures $\hat{y}$ based on the proposed model is compared and evaluated using the Variance-Accounted-For (VAF) and the Mean Magnitude of Relative Error (MMRE) [53].

$$VAF = [1 - \frac{var(y - \hat{y})}{var(y)}] \times 100\% \tag{12}$$

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{y_i} \tag{13}$$

Another evaluation criterion that we use in our study is the correlation coefficient R that can be calculated using the following equation.

$$R = \frac{\sum_{i=1}^{N}(y_i - \overline{y})(\hat{y}_i - \overline{\hat{y}})}{\sqrt{\sum_{i=1}^{N}(y_i - \overline{y})^2 \sum_{i=1}^{N}(\hat{y}_i - \overline{\hat{y}})^2}} \tag{14}$$

Finally we use the mean square error as the evaluation criterion in our convergence behavior analysis as shown in the following section.

$$MSE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{15}$$

V. Experiment Results

To develop our new technique for solving the problem of estimating the parameters of SRGM we used GWO MATLAB toolbox. We started by setting the number of search agents (grey wolves) and the maximum number of iterations for the experiment. From our experience we found that 30 agents and 50 iterations led to highly accepted results. The objective function gets the variables as a vector ($[x_1 x_2 ... x_n]$) and returns the objective value.

Our experiments explore the use of the GWO method to estimate the parameters of three software projects using three SRGMs. In each case, we estimate the model parameters for EXPM, POWM and DSSM models, generate the convergence curves using the GWO method and show the scattered plot.

A. Test/Debug Data 1

A real-time control application presented in [54], [55] is adopted as the first case study with a daily collected data. The real-time control application program has a size of 870 Kilo line of code (KLOC) of FORTRAN and a middle level language code. To estimate the model parameters b_0 and b_1 based on the GWO method, we needed to set up the search space. In our case, $b_0 \in [0, 500]$ and $b_1 \in [0, 1]$.

In Figure 1 (a), we show the actual and estimated accumulated failures curves for the EXPM, POWM and DSSM and the convergence behavior curves of the GWO process for the three developed models. A scattered plot of the three developed models is shown in Figure 1 (b).

Table I shows the estimated parameters for the SRGMs together with the model equations. In Table IV, we summarize the results of two evaluation criterion MMRE and VAF values for the three developed models EXPM, POWN and DSSM. In this case study, the DSSM model provided the best results in terms of VAF while the EXPM model's MMRE was the minimum in comparison to other POWM and DSSM.

B. Test/Debug Data 2

In the second case study, a real-time application [56] of a software system containing 200 modules of FORTRAN language was used to test our proposed methodology. The data consists of 111 measurements [55]. We ran the GWO to tune the parameters of EXPM, POWM and DSSM.

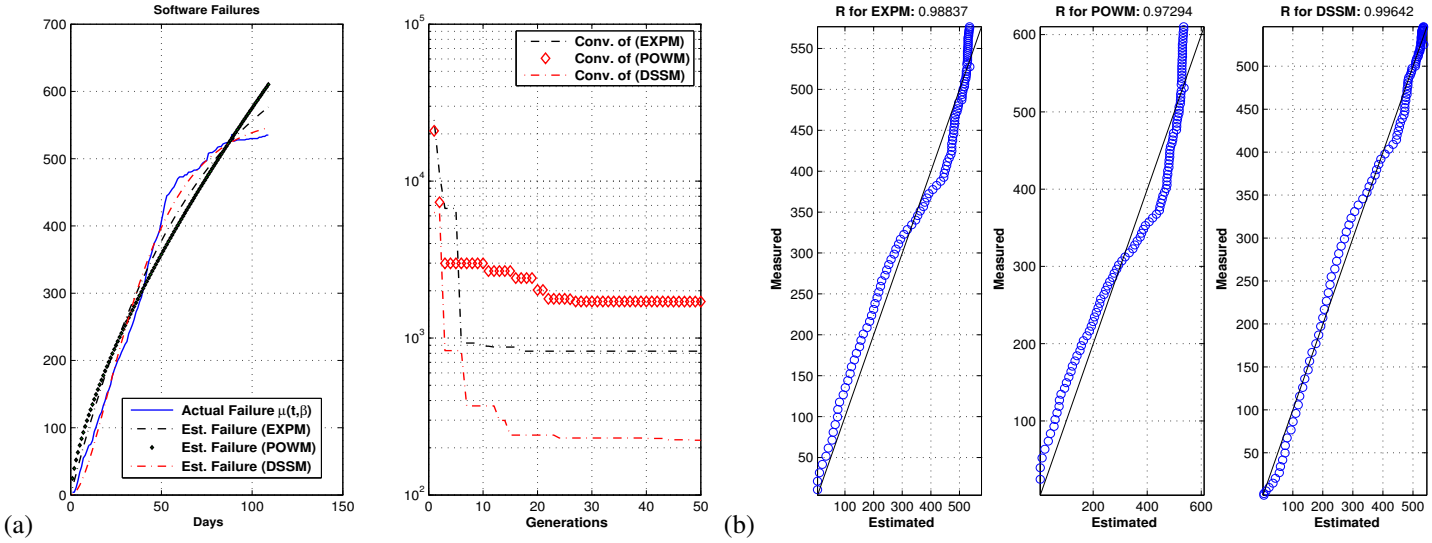

Fig. 1. (a) Actual and estimated failures and convergence curves for GWO (b) Scattered plot for the EXPM, POWM and DSSM using 109 Measurements

TABLE I
SRGMs WITH PARAMETER ESTIMATED USING GWO - 109 MEASUREMENTS

Exponential Model (EXPM)	$\mu(t;b)$ = 717.098 (1 - $e^{-0.01495539t}$)
Power Model (POWM)	$\mu(t;b)$ = 24.541 $t^{0.684974}$
Delayed S-Shaped Model (DSSM)	$\mu(t;b)$ = 562.3995 (1 - (1 + 0.04947323 t)$e^{-0.04947323t}$)

TABLE II
SRGMs WITH PARAMETER ESTIMATED USING GWO - 111 MEASUREMENTS

Exponential Model (EXPM)	$\mu(t;b)$ = 538.6468 (1 - $e^{-0.025683171t}$)
Power Model (POWM)	$\mu(t;b)$ = 30 $t^{0.625803}$
Delayed S-Shaped Model (DSSM)	$\mu(t;b)$ = 486.3256 (1 -(1 + 0.06691487 t)$e^{-0.06691487t}$)

In our case, $b_0 \in [0, 30]$ and $b_1 \in [0, 2]$. In Figure 2 (a), we show the actual and estimated accumulated failures curves for the EXPM, POWM and DSSM models and the convergence curves of the GWO process for the three developed models. A scattered plot of the three developed models is shown in Figure 2 (b).

Table II shows the estimated parameters for the SRGM models together with the model equations. The computed evaluation criterion are included in Table IV. Based on the developed experiments for this case, the results show that the DSSM model provided the best performance using the GWO tuned parameters as it has the minimum MMRE and the maximum VAF compared to other proposed models.

C. Test/Debug Data 3

In our third case study, we used a Test/Debug data set including 46 measurements as presented in [56]. We ran the GWO to find the best parameters to tune the EXPM, POWM and DSSM. In our case, $b_0 \in [0, 1000]$ and $b_1 \in [0, 1]$. In Figure 2 (a), we show the actual and estimated accumulated failures curves for the EXPM, POWM and DSSM models and the convergence curves of the GWO process for the three developed models. A scattered plot of the three developed models is shown in Figure 2 (b).

The estimated parameters for SRGMs are shown in Table III. In this case, the results show that the DSSM model was able to provide the best results in terms of MMRE while both the EXPM and the POWM models have better VAF values as shown in Table IV.

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed a GWO-based methodology to estimate the parameters of software reliability growth models (SRGMs). The estimated model parameters are used to predict the accumulated failures in a software system during the testing process. The problem is formulated for the GWO algorithm with the objective of minimizing the difference between the actual failures and the estimated accumulated failures.

Our methodology was employed to estimate the parameters of three adopted SRGMs: the exponential model, power model, and S-shaped model. Then the proposed models were applied to three real measured test/debug datasets. The results show that the proposed methodology is able to successfully estimate the parameters of SRGMs.

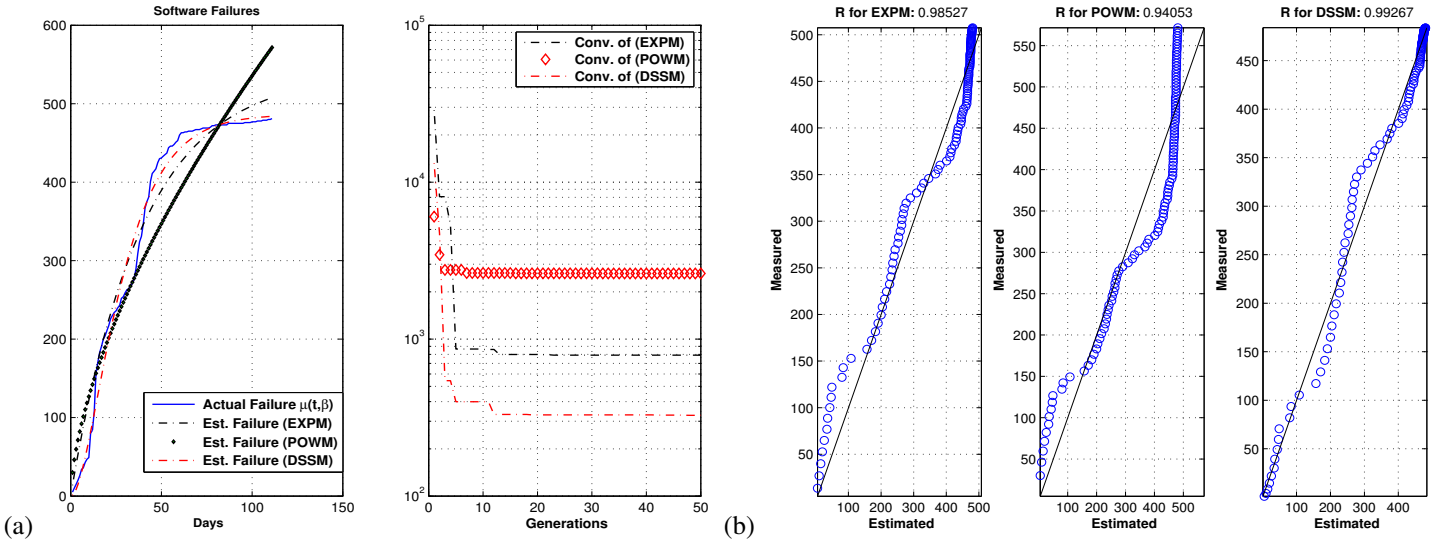

Fig. 2. (a) Actual and estimated failures and convergence curves for the GWO (b) Scattered plot for the EXPM, POWM and DSSM using 111 Measurements

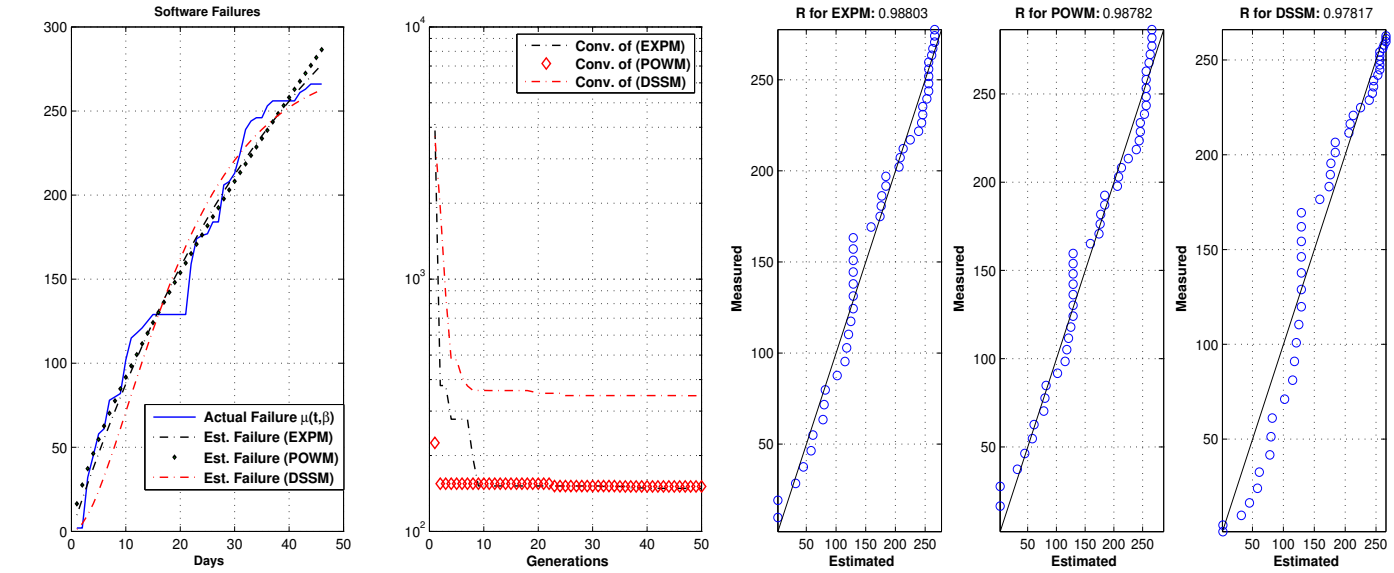

Fig. 3. (a) Actual and Estimated Accumulated failures and the convergence curves for the GWO (b) Scattered plot for the EXPM, POWM and DSSM using 46 Measurements

TABLE III
SRGMs WITH PARAMETER ESTIMATED USING GWO - 46 MEASUREMENTS

Exponential Model (EXPM)	$\mu(t;b)$ = 422.5453(1 - $e^{-0.02324815t}$)
Power Model (POWM)	$\mu(t;b)$ = 16.4506 $t^{0.746282}$
Delayed S-Shaped Model (DSSM)	$\mu(t;b)$ = 280.2617(1 - (1 + 0.09711093 t)$e^{-0.09711093t}$)

TABLE IV
EVALUATION RESULTS OF THE THREE MODELS USING GWO

	Test/Debug Data 1		Test/Debug Data 2		Test/Debug Data 3	
	MMRE	VAF%	MMRE	VAF%	MMRE	VAF%
EXPM	0.19027	97.347	0.19998	96.536	15.683	97.611
POWM	37.05	94.394	0.32297	88.343	22.736	97.566
DSSM	8.8572	99.268	0.072361	98.536	8.3919	94.701

For verification, a convergence behavior analysis was conducted. The results verify the effectiveness of the GWO algorithm to solve the problem with highly accepted performance. For future work, we plan to explore other techniques for modeling the software reliability growth based on other search algorithms in an effort to improve performance.

REFERENCES

[1] J. McCall, *Factors in Software Quality: Preliminary Handbook on Software Quality for an Acquisiton Manager*, vol. 1-3. General Electric, November 1977.

[2] M. W. Evans and J. J. Marciniak, *Software Quality Assurance and Management*. New York, USA: John Wiley and Sons, 1987.

[3] M. S. Deutsch and R. R. Willis, eds., *Software Quality Engineering, A Total Technical Management Approach, Ch.3*. Englewood Cliffs, NJ, USA: Prentice Hall, 1988.

[4] I. Sommerville, *Software Engineering: (Update) (8th Edition) (International Computer Science)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[5] J. Musa, "A theory of software reliability and its application," *IEEE Trans. Software Engineering*, vol. 1, pp. 312–327, 1975.

[6] J. Musa, A. Iannino, and K. Okumoto, *Software Reliability: Measurement, Prediction, Applications*. McGraw Hill, 1987.

[7] H. Pham, *Software Reliability*. Springer-Verlag, 2000.

[8] P. G. Bishop and R. Bloomfield, "Worst case reliability prediction on a prior estimate of residual defects," in *Proceedings of the 13th IEEE International Symposium on Software Reliability Engineering (ISSRE-2002)*, pp. 295–303, 2002.

[9] J. Musa, "Data analysis center for software: An information analysis center," *Western Michigan University Library, Kalamazoo, Michigan*, 1980.

[10] J. Musa, *Software Reliability Engineering: More Reliable Software, Faster and Cheaper*. Published Author House, 2004.

[11] J. Musa and L. A. Williams, "How should software reliability engineering be taught?," in *ISSRE*, p. 3, 2005.

[12] N. Karunanithi, D. Whitley, and Y. K. Malaiya, "Prediction of software reliability using connectionist models," *IEEE Trans. on Software Engineering*, vol. 18, no. 7, 1992.

[13] E. O. Costa, S. R. Vergilio, A. Pozo, and G. Souza, "Modeling software reliability growth with genetic programming," in *Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering*, ISSRE '05, (Washington, DC, USA), pp. 171–180, IEEE Computer Society, 2005.

[14] S. H. Aljahdali, D. Rine, and A. Sheta, "Prediction of software reliability: A comparison between regression and neural network non-parametric models," in *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, AICCSA '01, (Washington, DC, USA), pp. 470–, IEEE Computer Society, 2001.

[15] R. Kumar, K. Khatter, and A. Kalia, "Measuring software reliability: A fuzzy model," *SIGSOFT Softw. Eng. Notes*, vol. 36, pp. 1–6, Nov. 2011.

[16] A. Amin, L. Grunske, and A. Colman, "An approach to software reliability prediction based on time series modeling," *J. Syst. Softw.*, vol. 86, pp. 1923–1932, July 2013.

[17] J. Wang, Z. Wu, Y. Shu, Z. Zhang, and L. Xue, "A study on software reliability prediction based on triple exponential smoothing method (wip)," in *Proceedings of the 2014 Summer Simulation Multiconference*, SummerSim '14, (San Diego, CA, USA), pp. 61:1–61:9, Society for Computer Simulation International, 2014.

[18] M. K. Bhuyan, D. P. Mohapatra, and S. Sethi, "A survey of computational intelligence approaches for software reliability prediction," *SIGSOFT Softw. Eng. Notes*, vol. 39, pp. 1–10, Mar. 2014.

[19] J. Pati and K. K. Shukla, "A hybrid technique for software reliability prediction," in *Proceedings of the 8th India Software Engineering Conference*, ISEC '15, (New York, NY, USA), pp. 139–146, ACM, 2015.

[20] M. K. Bhuyan, D. P. Mohapatra, and S. Sethi, "Measures for predicting software reliability using time recurrent neural networks with back-propagation," *SIGSOFT Softw. Eng. Notes*, vol. 40, pp. 1–8, Sept. 2015.

[21] R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M. Y. Wong, "Orthogonal defect classification-a concept for in-process measurements," *IEEE Transactions on Software Engineering*, vol. 18, pp. 943–956, Nov 1992.

[22] N. E. Fenton and M. Neil, "A critique of software defect prediction models," *IEEE Trans. Softw. Eng.*, vol. 25, pp. 675–689, Sept. 1999.

[23] F. Deissenboeck, E. Juergens, K. Lochmann, and S. Wagner, "Software quality models: Purposes, usage scenarios and requirements," in *Proceedings of the Seventh ICSE Conference on Software Quality*, WOSQ'09, (Washington, DC, USA), pp. 9–14, IEEE Computer Society, 2009.

[24] W.-L. Wang, D. Pan, and M.-H. Chen, "Architecture-based software reliability modeling," *J. Syst. Softw.*, vol. 79, pp. 132–146, Jan. 2006.

[25] H. Hu, C.-H. Jiang, K.-Y. Cai, W. E. Wong, and A. P. Mathur, "Enhancing software reliability estimates using modified adaptive testing," *Inf. Softw. Technol.*, vol. 55, pp. 288–300, Feb. 2013.

[26] S. H. Aljahdali and M. E. El-Telbany, "Genetic algorithms for optimizing ensemble of models in software reliability prediction," *Artificial Intelligence and Machine Learning (AIML)*, vol. 8, pp. 5–13, 6 2008.

[27] S. Yamada, *Software Reliability Modeling: Fundamentals and Applications*. Springer Publishing Company, Incorporated, 2013.

[28] N. R. Barraza, "A parametric empirical bayes model to predict software reliability growth," *Procedia Computer Science*, vol. 62, pp. 360 – 369, 2015. Proceedings of the 2015 International Conference on Soft Computing and Software Engineering (SCSE'15).

[29] L. K. Singh, G. Vinod, and A. K. Tripathi, "Early prediction of software reliability: A case study with a nuclear power plant system," *IEEE Computer*, vol. 49, no. 1, pp. 52–58, 2016.

[30] S. Kundu, T. K. Nayak, and S. Bose, *Statistical Models and Methods for Biomedical and Technical Systems*, ch. Are Nonhomogeneous Poisson Process Models Preferable to General-Order Statistics Models for Software Reliability Estimation?, pp. 137–152. Boston, MA: Birkhäuser Boston, 2008.

[31] P. Kapur, D. Goswami, A. Bardhan, and O. Singh, "Flexible software reliability growth model with testing effort dependent learning process," *Applied Mathematical Modelling*, vol. 32, no. 7, pp. 1298 – 1307, 2008.

[32] K.-Y. Cai, D.-B. Hu, C.-G. Bai, H. Hu, and T. Jing, "Does software reliability growth behavior follow a non-homogeneous poisson process," *Inf. Softw. Technol.*, vol. 50, pp. 1232–1247, Nov. 2008.

[33] Z. Wang, J. Wang, and X. Liang, "Non-parametric estimation for nhpp software reliability models," *Journal of Applied Statistics*, vol. 34, no. 1, pp. 107–119, 2007.

[34] S. Aljahdali, A. F. Sheta, and D. Rine, "Predicting accumulated faults in software testing process using radial basis function network models," in *Proceedings of the ISCA 17th International Conference Computers and Their Applications, April 4-6, 2002, Canterbury Hotel, San Francisco, California, USA*, pp. 26–29, 2002.

[35] H. Zeng and D. Rine, "Estimation of software defects fix effort using neural networks," in *28th International Computer Software and Applications Conference (COMPSAC 2004), Design and Assessment of Trustworthy Software-Based Systems, 27-30 September 2004, Hong Kong, China, Workshop Papers*, pp. 20–21, 2004.

[36] S. Wagner, "A bayesian network approach to assess and predict software quality using activity-based quality models," in *Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, PROMISE '09, (New York, NY, USA), pp. 6:1–6:9, ACM, 2009.

[37] K. Jeet, R. Dhir, and H. Verma, "A comparative study of bayesian and fuzzy approach to assess and predict maintainability of the software using activity-based quality model," *SIGSOFT Softw. Eng. Notes*, vol. 37, pp. 1–9, May 2012.

[38] A. Sheta, "Reliability growth modeling for software fault detection using particle swarm optimization," in *Proceedings of the 2006 IEEE Congress on Evolutionary Computation (CEC2006)*, pp. 3071–3078, 2006.

[39] A. Sheta, "Parameter estimation of software reliability growth models by particle swarm optimization," *Artificial Intelligence and Machine Learning (AIML)*, vol. 7, pp. 55–61, 9 2007.

[40] M. Xie, "Software reliability models - past, present and future," *In N. Limnios and M. Nikulin (Eds). Recent Advances in Reliability Theory: Methodology, Practice and Inference*, pp. 323–340, 2002.

[41] S. Yamada, "Software reliability models and their applications: A survey," in *International Seminar on Software Reliability of Man-Machine Systems - Theories Methods and Information Systems Applications - August 17-18, Kyoto University, Kyoto, Japan*, 2000.

[42] P. B. Moranda, "Predictions of software reliability during debugging," in *Proceedings of Annual Reliability and Maintainability Symposium*, pp. 327–332, 1975.

[43] A. Geol and K. Okumoto, "Time-dependent error-detection rate model

for software reliability and other performance measures," *IEEE Trans. Reliability*, vol. 28, pp. 206–211, 1979.

[44] L. H. Crow, "Reliability for complex repairable systems," *Reliability and Biometry, SIAM*, pp. 379–410, 1974.

[45] G.-Y. PARK and S. C. JANG, "A software reliability estimation method to nuclear safety software," *Nuclear Engineering and Technology*, vol. 46, no. 1, pp. 55 – 62, 2014.

[46] S. Yamada, M. Ohba, and O. S., "S-Shaped reliability growth modeling for software error detection," *IEEE Trans. Reliability*, pp. 475–478, 1983.

[47] S. Yamada, M. Ohba, and O. S., "S-Shaped software reliability growth models and their applications," *IEEE Trans. Reliability*, pp. 289–292, 1984.

[48] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.

[49] M. H. Sulaiman, Z. Mustaffa, M. R. Mohamed, and O. Aliman, "Using the gray wolf optimizer for solving optimal reactive power dispatch problem," *Appl. Soft Comput.*, vol. 32, pp. 286–292, July 2015.

[50] S. Mirjalili, "How effective is the grey wolf optimizer in training multi-layer perceptrons," *Applied Intelligence*, vol. 43, pp. 150–161, July 2015.

[51] N. Jayakumar, S. Subramanian, S. Ganesan, and E. B. Elanchezhian, "Combined heat and power dispatch by grey wolf optimization," *International Journal of Energy Sector Management*, vol. 9, no. 4, pp. 523–546, 2015.

[52] E. Emary, H. M. Zawbaa, C. Grosan, and A. E. Hassenian, *Afro-European Conference for Industrial Advancement: Proceedings of the First International Afro-European Conference for Industrial Advancement AECIA 2014*, ch. Feature Subset Selection Approach by Gray-Wolf Optimization, pp. 1–13. Cham: Springer International Publishing, 2015.

[53] M. Shin and A. L. Goel, "Empirical data modeling in software engineering using radial basis functions," *IEEE Transactions on Software Engineering*, vol. 26, no. 6, pp. 567–576, 2000.

[54] T. Minohara and Y. Tohma, "Parameter estimation of hyper-geometric distribution software reliability growth model by genetic algorithms," in *Proceedings of the 6th International Symposium on Software Reliability Engineering*, pp. 324–329, 1995.

[55] A. Sheta, "Reliability growth modeling for software fault detection using particle swarm optimization," in *2006 IEEE Congress on Evolutionary Computation, Sheraton, Vancouver Wall Centre, Vancouver, BC, Canada, July 16-21, 2006.*, pp. 10428–10435, 2006.

[56] Y. Tohman, K. Tokunaga, S. Nagase, and M. Y., "Structural approach to the estimation of the number of residual software faults based on the hyper-geometric distribution model," *IEEE Trans. on Software Engineering*, pp. 345–355, 1989.