

Parameter Estimation of Software Reliability Growth Models by Particle Swarm Optimization

Alaa Sheta

Information Technology Department

Prince Abdullah Bin Ghazi Faculty of Science and Information Technology,

Al-Balqa Applied University

Al-Salt, Jordan,

sheta@bau.edu.jo

<http://www.bau.edu.jo>

Abstract

Building software reliability growth models (SRGM) for predicting software reliability represents a challenge for software testing engineers. Being able to predict the number of faults (failure) in the software during development and testing processes helps significantly in specifying/computing the software release day and in managing project resources (i.e. people and money). In this paper, we explore the use of Particle Swarm Optimization (PSO) algorithm to estimate SRGM parameters. The proposed method shows significant advantages in handling variety of modeling problems such as the exponential model (EXPM), power model (POWM) and Delayed S-Shaped model (DSSM). PSO algorithm will be used to estimate the parameters of the well known SRGM. Detailed results and analysis are provided showing the potential advantages of using PSO in solving this problem.

Keywords: *Particle Swarm Optimization, Software Reliability Growth Modeling, Software Testing.*

1 Introduction

Software reliability is defined according to [21] as *the probability of failure free operation of a computer program in a specified environment for a specified period of time*. Failure process modeling represents a challenge because of the various nature of faults discovered and the methodologies to be used in order to isolate the faults [22, 4]. Unstable and unreliable software system can affect people life. In the following, we give few examples. In [26], author stated that, "In June 4th 1996, a total failure of the Ariane 5 launcher on its

maiden flight was reported. Software failure occurred when a process for converting a 64-bit floating point number to a signed 16-bit integer was running. This operation caused a number overflow. Unfortunately, the backup software was just a copy of the original system. It behaved in exactly the same way. The mission failure was a result of a software failure." The problem of power shutdown of USS Yorktown problem is yet another software failure problem. The problem happen when a sailor mistakenly typed 0 in a field of the kitchen inventory application. Subsequent division by this field caused an arithmetic exception, which propagated through the system, crashed all LAN consoles and remote terminal units, and lead to power shutdown for about 3 hours. Another failure was when a bug in the Pentium processor in 1994 has happened. It was found that the omission of five entries in a table of 1,066 values (part of the chips circuitry) used by a speedy algorithm known as SRT division. This problem cost Intel company about \$500 million. This list can be extended.

Interest in using evolutionary computation and soft computing techniques, such as Genetic Algorithms (GAs), Genetic Programming (GP), Evolutionary strategies (ESs), Artificial Neural Networks (ANN), Fuzzy Logic (FL), and Particle Swarm Optimization (POS), to solve software engineering problems expanded in the recent years. Estimation of the COCOMO model parameters using Genetic Algorithms (GAs) for NASA Software Projects were provided in [24]. Predicting accumulated faults during the software testing process using parametric and non-parametric models were explored in many articles [1, 2, 25]. In [34], author provided a strategic solution for estimating software defect fix effort using self-organizing neural network. Fuzzy logic and neural networks were used in software engineering project management [7, 14].



In this paper, we explore the use of PSO to predict the faults during the software testing process using software faults historical data. Detailed results are provided to explore the advantages of using PSO in solving this problem. In section 3, we provide an overview of various SRGM which we will use in this article. The PSO algorithm is provided in section 4. Detailed experiments are provided in section 5. Then, a conclusion and future work is provided.

2 The SRGM

In the past three decades, hundreds of models were introduced to estimate the reliability of software systems [31, 33]. The issue of building growth models was the subject of many research work [16, 20] which helps in estimating the reliability of a software system before its release to the market. Serious application such as weapon systems and NASA space shuttle applications were explored [5, 8, 23].

2.1 Exponential Model (EXPM)

The exponential model was first provided in [18, 19].

$$\begin{aligned}\mu(t; \beta) &= \beta_0(1 - e^{-\beta_1 t}) \\ \lambda(t; \beta) &= \beta_0 \beta_1 e^{-\beta_1 t}\end{aligned}\quad (1)$$

$\mu(t; \beta)$ and $\lambda(t; \beta)$ represent the mean failure function and the failure intensity function, respectively. The parameters β_0 is the initial estimate of the total failure recovered at the end of the testing process (i.e. v_0). β_1 represents the ratio between the initial failure intensity λ_0 and total failure v_0 . Thus, $\beta_1 = \lambda_0/v_0$.

It is important to realize that:

$$\lambda(t; \beta) = \frac{\partial \mu(t; \beta)}{\partial t}\quad (2)$$

This model is known as a finite failure model. Many traditional techniques exists for the estimation of the model parameters β_0 and β_1 . They include the least square estimation. This technique has many problems related to the estimation accuracy and it needs a large number of measurements so that it can provide a good parameter estimate.

2.2 Power Model (POWM)

The Power model was provided in [6]. The equations which govern the relationship between the time t and both $\mu(t; \beta)$ and $\lambda(t; \beta)$ are:

$$\begin{aligned}\mu(t; \beta) &= \beta_0 t^{\beta_1} \\ \lambda(t; \beta) &= \beta_0 \beta_1 t^{\beta_1 - 1}\end{aligned}\quad (3)$$

The model objective is to compute the reliability of a hardware system during testing process. The model is based on the non-homogeneous Poisson process model.

2.3 Delayed S-Shaped Model (DSSM)

This model describes the software reliability process as a delayed S-shaped model [32]. This model is also a finite failure model. The system equation for $\mu(t; \beta)$ and $\lambda(t; \beta)$ are:

$$\begin{aligned}\mu(t; \beta) &= \beta_0(1 - (1 + \beta_1 t)e^{-\beta_1 t}) \\ \lambda(t; \beta) &= \beta_0 \beta_1^2 t^{-\beta_1 t}\end{aligned}\quad (4)$$

The model represents a learning process since some improvement was added to the exponential model based the growing experience of the project team.

3 Particle Swarm Optimization

PSO is a population based search algorithm. PSO was inspired from natural animal behavior [11, 9, 10, 13]. The population contains set of particles each of which represents a solution for a given optimization problem. These particles are normally initialized randomly. During the PSO process, each particle, based on given evaluation criterion; update its own position with a certain velocity. The velocity is computed based on both the best experience of the particle itself and that of the entire population. This update process is repeated for number of generations. The update process stops either when the objective (i.e. solution of the problem) is achieved or when the maximum number of generations reached.

PSO has been effectively used to solve variety of software engineering problems with better results than GAs. In [30], authors explored the idea of using an evolutionary model to automatically generating test cases that can provide high structural code coverage. A set of experiments where provided. They included 25 small artificial test objects and 13 more complex industrial test objects taken from various development projects. Author reported the success of PSO to perform in a better way than GAs for most test codes in terms of effectiveness and efficiency. A preliminary results on parameter estimation of SRGM using PSO was presented in [25]. The sensitivity of the tuning parameters (i.e the swarm size and velocity update) of PSO was also studied in various articles [29, 28].

3.1 The Algorithm

The PSO algorithm can be described in three stages. They are: 1) initial generation of particles position and velocities; 2) updating velocities; 3) updating positions.

Step 1: Consider a swarm has "p" particles. The positions x_k^i and velocities v_k^i of the initial swarm of particles will be initialized randomly in a specific domain. The upper and lower bounds for each particle are presented in Equation 5.

$$x_0^i = x_{min} + r(x_{max} - x_{min})$$



$$v_0^i = \frac{x_{min} + r(x_{max} - x_{min})}{\delta t} \quad (5)$$

Step 2: The velocity of each particle is updated according to Equation 6.

$$v_{k+1}^i = \alpha_k v_k^i + \gamma_1 r_1 (p_k^i - x_k^i) + \gamma_2 r_2 (g_k - x_k^i) \quad (6)$$

where k represents the instance in time.

- α_k is called the inertia weight; It is used a control parameter for the PSO process. It shows how the history of the process will affect the future. In other word, too large value for α_k mean a large memory of the algorithm. It is important to make a balance between global and local exploration of the algorithm (i.e. exploration of various areas on the search space and exploitation of the neighborhood).
- γ_1, γ_2 are two positive constant called cognitive and social parameters, respectively. Fine tuning for the parameters γ_1, γ_2 will help to achieve faster convergence of the PSO.
- The parameters r_1, r_2 are two random numbers having values between $[0,1]$. Their role is to keep population diversity.
- p_k^i is defined as the best location found by particle i up to the instance of time k .
- g_k is defined as the best global position found among all particle in the swarm up to instance k .

Step 3: Finally, the position is updated as given in Equation 7:

$$x_{k+1}^i = x_k^i + v_{k+1}^i \delta t \quad (7)$$

The fitness for the locations p_k^i, g_k is defined as f_{best}^i and f_{best}^g , respectively. Our objective is to minimize the difference between f_{best}^i and f_{best}^g such that no further improvement is introduced. It normally takes few hundred to few thousands iterations till convergence achieved. The algorithm stops when the error minimum. The pseudo code of the PSO procedure was presented in [12, 15] and is given in Figure 1.

4 Experiment Setup

To solve the parameter estimation problem for software reliability modeling we used the PSO toolbox under MATLAB [3]. Our objective is to estimate the parameters such that the error difference, between the actual fault and the estimated faults based the parameter estimated using PSO, is minimal. The evaluation criterion to measure the performance of the developed

```

For each particle
    Initialize particle
For each particle
    Calculate fitness value
    If the fitness value is better than the best fitness
        value (pbest) in history
        set current value as the new pbest
End
Choose the particle with the best fitness value of
all the particles as the gbest
For each particle
    Calculate particle velocity according to Eq. 5
    Update particle position according to Eq. 6
End
Continue while maximum iterations or minimum
error criteria is not attained
    
```

Figure 1: The Pseudo code of the PSO procedure

Table 1: The tuning parameters for the PSO

Operator	Value
Acceleration constant	[2.1,2.1]
Inertia Weight	[0.9,0.6]
Maximum no. of Iteration	1000
Maximum Velocity	300
No. of Particles	2
Domain of search for a	[-1000,1000]
Domain of search for b	[-1,1]

PSO models is selected to be the route mean square of the error:

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

N represents the number of measurements used for estimating the model parameters. The performance of the PSO algorithm is evaluated by computing the objective evaluation criterion in terms of the fitness. The tuning parameters and search space for PSO is given in Table 1. In the following sections, we will provide the results of a three real measured test/debug data set where our proposed approach was applied.

4.1 Test/Debug Data 1

The data set which contain real measured test/debug date of a real-time control application is presented in [17, 25]. The data set given includes the reading of the measured faults (x_k), the accumulated faults (y_k) and the the number of test workers (tw_k) involved in the testing process. The developed program size was 870 Kilo line of code (KLOC) of FORTRAN and a middle level language. The test/debug data was measured day by day. The day represents the test instance. To build a software reliability model, we used 70% of the



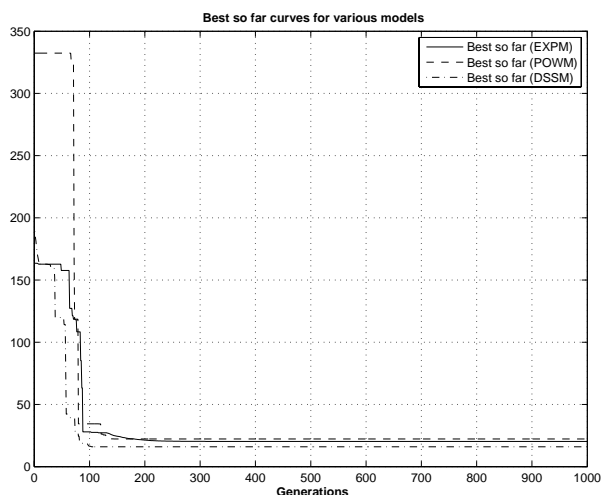


Figure 2: Best so far curve for PSO convergence - fitness function RMSE - 109 Measurements

Table 3: Computed RMSE for Test/Debug Data 1

Model	RMSE-Training	RMSE-Testing
EXPM	20.2565	119.4374
POWM	22.2166	152.9372
DSSM	15.9237	26.3015

measurements to estimate the model parameters. The rest of the dataset was used for prediction/validation of the developed model.

The evolutionary process starts by running the PSO with the tuning parameters given in Table 1. We run PSO for 1000 iteration to obtain the parameters in the three cases under study. The convergence process was recorded by keeping an eye on the best individual after each generation.

In Figure 2, we show the best so far curves of the evolutionary process for the three developed models. The actual and accumulated faults (failures) curves for the exponential model, power model and delayed S-shaped model are shown in Figure 3.

The estimated parameters, for the software reliability growth models, are given with the model equations in Table 2. The computed RMSE in both training and testing (validation) cases are shown in Table 3. In this case, we found that the Delayed S-Shaped model was able to provide the best results using the PSO tuned parameters. The model error was the minimum compared to other proposed models.

4.2 Test/Debug Data 2

A field report data was developed to measure system faults during testing in a real-time application [27]. The software system consists of 200 modules with each having one kilo line of code of FORTRAN. We run the PSO to find the best parameters to tune the exponential model, power model and Delayed S-Shaped

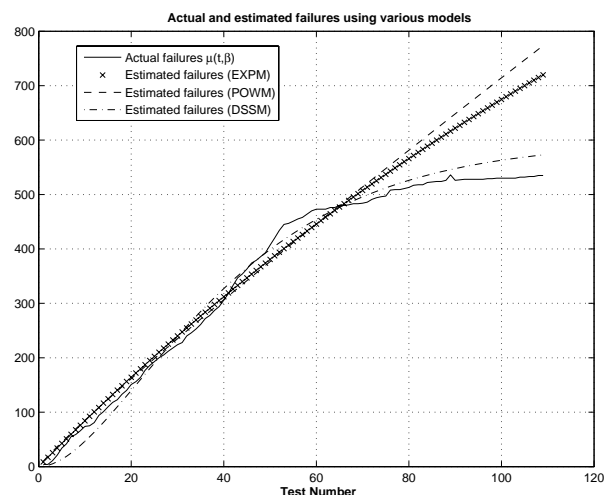


Figure 3: Actual and Accumulated failures for the EXPM, POWM and DSSM using PSO - 109 Measurements

Table 5: Computed RMSE for Test/Debug Data 2

Model	RMSE-Training	RMSE-Testing
EXPM	24.9899	80.8963
POWM	32.3550	149.9684
DSSM	20.8325	17.0638

model. A Test/Debug data set of 111 measurements presented in [25] was used for our experiments.

The best so far curves of the convergence process for the swarm process is presented in Figure 4. The estimated parameters for the developed SRGM are given in Table 4. The actual and accumulated faults (failures) curves for the EXPM, POWM and DSSM are shown in Figure 5. Based the developed experiments, the Delayed S-Shaped model provided the minimum RMSE among other models. The computed RMSE over both training and testing (validation) data are shown in Table 5. The results show that the DSSM is performing very well in this case.

4.3 Test/Debug Data 3

A Test/Debug data set has 46 measurements was presented in [27]. The number of measurements collected during testing process is small. This represents a difficulty for traditional parameter estimation techniques. It is sometimes difficult to correctly estimate model parameters using small number of measurements.

The estimated parameters for the software reliability growth models are given with the model equations in Table 7. The power model and the Delayed S-Shaped model provided good results although the power model was slightly better in the case of training.

In Figure 6, we show the best so far curves of the swarm evolutionary process for the three developed



Table 2: Reliability Growth Model with Parameter Estimated using PSO - 109 Measurements

Exponential Model (EXPM)	$\mu(t; \beta) = 1670.9987(1 - e^{-0.0051702t})$
Power Model (POWM)	$\mu(t; \beta) = 10.1655 t^{0.92338}$
Delayed S-Shaped Model (DSSM)	$\mu(t; \beta) = 596.729 (1 - (1 + 0.045927 t)e^{-0.045927t})$

Table 4: Reliability Growth Model with Parameter Estimated using PSO - 111 Measurements

Exponential Model (EXPM)	$\mu(t; \beta) = 695.7171(1 - e^{-0.017t})$
Power Model (POWM)	$\mu(t; \beta) = 21.7567 t^{0.73627}$
Delayed S-Shaped Model (DSSM)	$\mu(t; \beta) = 502.571(1 - (1 + 0.0635 t)e^{-0.0635t})$

Table 6: Computed RMSE for Test/Debug Data 3

Model	RMSE-Training	RMSE-Testing
EXPM	12.8925	13.6094
POWM	11.9446	14.0524
DSSM	18.5807	47.4036

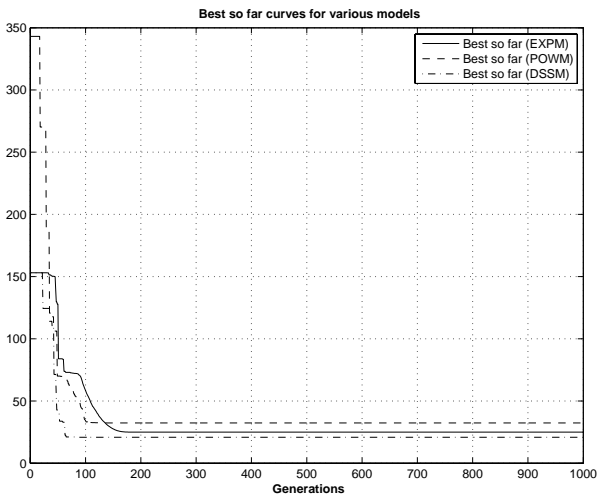


Figure 4: Best so far curve for PSO convergence - fitness function RMSE - 111 Measurements

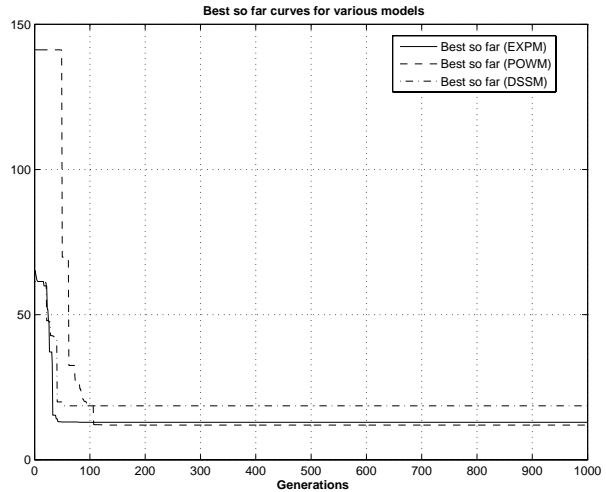


Figure 6: Best so far curve for PSO convergence - fitness function RMSE - 46 Measurements

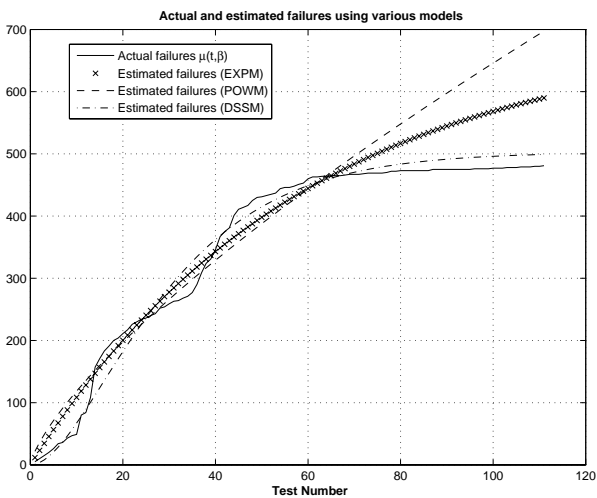


Figure 5: Actual and Accumulated failures for the EXPM, POWM and DSSM using PSO - 111 Measurements

models. The actual and accumulated faults (failures) curves for the exponential model, power model and delayed S-Shaped model are shown in Figure 7. The computed RMSE in both training and testing cases are shown in Table 6.

5 Conclusion and Future Work

In this paper, we explored the use of Particle Swarm Optimization (PSO) technique to estimate the parameters of software reliability models (i.e. exponential model, power model and S-shaped model). The estimated model parameters were used to predict the faults in a software system during the testing process. The PSO method shows significant advantages in handling the modeling problem for the exponential



Table 7: Reliability Growth Model with Parameter Estimated using PSO - 46 Measurements

Exponential Model (EXPM)	$\mu(t; \beta) = 382.4057(1 - e^{-0.0261t})$
Power Model (POWM)	$\mu(t; \beta) = 14.9701 t^{0.7760}$
Delayed S-Shaped Model (DSSM)	$\mu(t; \beta) = 216.3715 (1 - (1 + 0.1360 t)e^{-0.1360t})$

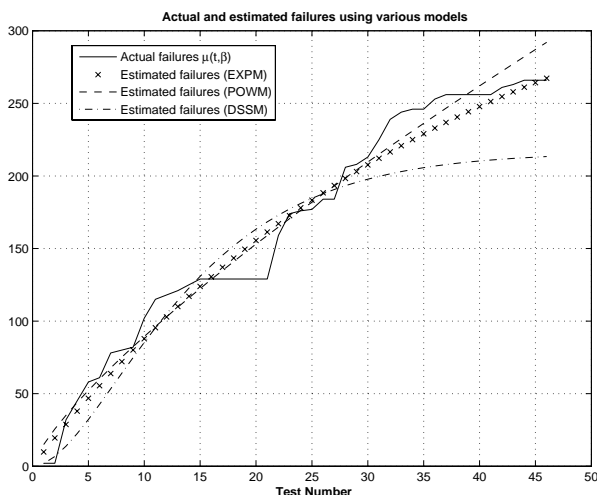


Figure 7: Actual and Accumulated failures for the EXPM, POWM and DSSM using PSO - 46 Measurements

model, the power model and the Delayed S-Shaped model. Our future work include the exploration of using Genetic Programming in developing a polynomial structure model which can provide an advanced relationship between $\mu(t; \beta)$ and $\lambda(t; \beta)$ to better model the software reliability prediction process. GP can build a polynomial structure with set of parameters which can provide an accurate estimate of the software faults.

References

- [1] Sultan Aljahdali, David Rine, and Alaa Sheta. Prediction of software reliability: A comparison between regression and neural network non-parametric models. In *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2001)*, Beirut, Lebanon, pages 470–473, 2001.
- [2] Sultan Aljahdali, Alaa Sheta, and David Rine. Predicting accumulated faults in software testing process using radial basis function network models. In *17th International Conference on Computers and Their Applications (CATA), Special Session on Intelligent Software Reliability*, San Francisco, California, USA, 2002.
- [3] Brian Birge. *PSOt, Particle Swarm Optimization Toolbox for Matlab*. Submitted to the MATLAB Central, April, 22, 2005.
- [4] 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)*, pages 295–303, 2002.
- [5] P. Carnes. Software reliability in weapon systems. In *Eighth International Symposium on Software Reliability Engineering (ISSRE '97)*, 1997.
- [6] L. H. Crow. Reliability for complex repairable systems. *Reliability and Biometry*, SIAM, pages 379–410, 1974.
- [7] A. C. Hodgkinson and P. W. Garratt. A neuro-fuzzy cost estimator. In *Proceedings of the Third Conference on Software Engineering and Applications*, pages 401–406, 1999.
- [8] T. Keller and N. Schneidewind. Successful application of software reliability engineering for the NASA space shuttle. In *Eighth International Symposium on Software Reliability Engineering (ISSRE '97)*, 1997.
- [9] J. Kennedy. The particle swarm: Social adaptation of knowledge. In *Proceedings of the 1997 International Conference on Evolutionary Computation*, pages 303–308. IEEE Service Center, Piscataway, NJ, 1997.
- [10] J. Kennedy. The behavior of particles. *Evolutionary Programming VII*, pages 581–587, 1998.
- [11] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948. Piscataway, NJ, USA, 1995.
- [12] J. Kennedy and R. C. Eberhart. Particle swarm optimization. *Proceedings of IEEE International Conference on Neural Networks (Perth, Australia)*, IEEE Service Center, Piscataway, NJ, 5(3):1942–1948, 1995.
- [13] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.
- [14] S. Kumar, B. A. Krishna, and P.S. Satsangi. Fuzzy systems and neural networks in software engineering project management. *Journal of Applied Intelligence*, 4:31–52, 1994.



- [15] Yong ling Zheng, Long hua Ma, Li yan Zhang, and Ji xin Qian. Empirical study of particle swarm optimizer with an increasing inertia weight. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation CEC2003*, pages 221–226, Canberra, 8-12 December 2003. IEEE Press.
- [16] M. R. Lyu. *Handbook of Software Reliability Engineering*. IEEE Computer Society Press, McGraw Hill, 1996.
- [17] 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*, pages 324–329, 1995.
- [18] P. B. Moranda. Predictions of software reliability during debugging. In *Proceedings of Annual Reliability and Maintainability Symposium*, pages 327–332, 1975.
- [19] John Musa. A theory of software reliability and its application. *IEEE Trans. Software Engineering*, 1:312–327, 1975.
- [20] John Musa. *Software Reliability Engineering: More Reliable Software, Faster and Cheaper*. Published AuthorHouse, ISBN 0079132715, 2004.
- [21] John Musa, A. Iannino, and K. Okumoto. *Software Reliability: Measurement, Prediction, Applications*. McGraw Hill, 1987.
- [22] H. Pham. *Software Reliability*. Springer-Verlag, 2000.
- [23] N. F. Schneidewind and Ted W. Keller. Applying reliability models to the space shuttle. *IEEE Transactions on Software Engineering*, pages 28–33, 1992.
- [24] Alaa Sheta. Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects. *Journal of Computer Science, USA*, 2(2):118–123, 2006.
- [25] Alaa 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*, pages 10428–10435, 2006.
- [26] Ian Sommerville. *Software Engineering, 6th Edition*. Pearson Education Limited, England, 2001.
- [27] Y. Tohman, K. Tokunaga, S. Nagase, and Murata 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*, pages 345–355, 1989.
- [28] F. van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.
- [29] F. van den Bergh and A.P. Engelbrecht. Effects of swarm size on cooperative particle swarm optimization. In *GECCO 2001, (San Francisco, USA)*, 2001.
- [30] Andreas Windisch, Stefan Wappler, and Joachim Wegener. Applying particle swarm optimization to software testing. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1121–1128, New York, NY, USA, 2007. ACM Press.
- [31] 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*, pages 323–340, 2002.
- [32] S. Yamada, M. Ohba, and Osaki S. S-Shaped software reliability growth models and their applications. *IEEE Trans. Reliability*, pages 289–292, 1984.
- [33] Shigeru 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.
- [34] Hui Zeng and David Rine. A neural network approach for software defects fix effort estimation. In *Proceedings of the Eighth IASTED International Conference Software Engineering and Applications*, pages 513–517, 2004.

