iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 5 **Issue:** IX **Month of publication:** September 2017

DOI: http://doi.org/10.22214/ijraset.2017.9017

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue IX, September 2017- Available at www.ijraset.com

A Comprehensive Literature Survey on the Use of Particle Swarm Optimization Technique for Software Effort Estimation

Mandeep Kaur[1]
[1] *Department of Computer Science, Khalsa College for Women, Civil Lines, Ludhiana.*

Abstract: ***Software effort estimation is the process of finding out the amount of effort that would be required to develop software. It forms the basis for planning out the various software development activities, estimating the resources-time and manpower, that will be required in developing a software. It is measured in person-months. The quality and accuracy of the estimated effort greatly affects the success of a project. One of the computational intelligence techniques that have been widely used for optimizing a non linear and multidimensional problem is particle swarm optimization technique which is inspired by the social behavior of flock of birds and school of fish. Thus particle swarm optimization can be used in optimizing various effort estimation methods. This paper represents the comprehensive survey on use of particle swarm optimization for effort estimation from the year 2008 to 2016.***
Keywords: Particle Swarm Optimization (PSO), Effort Estimation, COCOMO, LOC, Function Points.

I. INTRODUCTION

With the increase in the competition and complexity in the software industry, it has become important to effectively estimate the effort, cost and time that would be required to complete a software project. These estimates are based on available documents, past records or experiences, assumptions, certain identified risks, etc. For this, many organizations make use of software metrics in the process of software management. One of the most important elements of software metrics is size which further helps in estimating effort, cost, time duration, and scheduling. Effort estimation is a process of software project development which must be done before the beginning of software programming. The accurate effort estimation of the software tries to make the execution procedure of the software development teams and allocation of the resources take place according to the software engineering factors and the project manager makes corrections facing any inconformity in the execution procedure. So, the project manager executes the required techniques for the activities to meet the needs taking into consideration the estimation effort. Also using the project records is very effective in success of the software projects and the estimation could be done more reliably.

II. SOFTWARE METRICS

The two most widely used software metrics are: LOC (Lines of Code) and Function Points [1]. A Line of Code measures the project size by counting the number of source instructions of a program. Here the comments and header lines are not considered. To measure the size of the project LOC needs to be determined before beginning the project, this is what forms the basis for effort estimation. To effectively evaluate the LOC the project managers divide the problem into smaller modules with each module further divided into sub modules using the bottom –up approach the total size of the project is estimated. The widely used COCOMO (Constructive Cost Model) series of models invented by Dr. Barry Boehm [2] for effort estimation makes use of KLOC (Kilo Lines of Code) as software metric.

$$\text{Effort}=a\,(\text{KLOC})^{b} \qquad (1.1)$$

A. Software projects are classified based on the complexity of the project into three categories. They are

1) Organic
2) Semidetached
3) Embedded [3]

B. However there are a few shortcomings of LOC

1) The size of the problem can vary with respect to individual coding style. Hence, even for the same problem, different programmers might come up with programs having different LOC counts [2].
2) LOC focuses merely on coding and does not consider the total effort needed to specify, design, code, test, etc.

3) The LOC count can be accurately computed only after the code has been fully developed.
4) LOC will change depending upon the coding language opted.

The second software metric is function point. Function points measure the software size in terms of the functionality provided to user. They are independent of the language, tools or methodologies used for implementation, i.e. they do not take into consideration the programming languages, database management system or processing hardware. According to IFPUG, function points are calculated on the basis of Unadjusted Function Point (UFP) and Value Adjustment Factor (VAF) where UFP is evaluated on the basis of external input, output, enquiry, files and interfaces [4][5]. Different function point based effort estimation models were developed by Albrecht-Gaffney[6], Kemerer [7] and Matson, Barrett and Mellichamp[8].

$$VAF = 0.65 + [(\sum_{i=1}^{14} Ci) * 0.01] \quad (1.2)$$

Where, i = each GSC, from 1 to 14.
Ci = degree of influence for each GSC (General System Characteristic).
∑= is summation of all 14 GSC's.
The degree of influence of each characteristic is determined as a rating on a scale of 0 to 5 as defined below.
0 = Not present or no Influence
1 = Incidental influence
2 = Moderate influence
3 = Average influence
4 = Significant influence
5 = Strong influence throughout
Table 1.1 specifies the functional complexities for each of the function types which can be used to evaluate the unadjusted function point values.

Table 1.1: UFP Calculation

Function Type	Weight Value by Functional Complexity			Total
	Low	Average	High	
External Input	__*3	__*4	__*6	
External Output	__*4	__*5	__*7	
External Enquiry	__*3	__*4	__*6	
External Interface File	__*5	__*7	__*10	
Internal Logical File	__*7	__*10	__*15	
Total Number of Unadjusted Function Points =				

Finally the function points are evaluated as

$$FP = UFP * VAF \quad (1.3)$$

III. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a computational intelligence method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality [9]. PSO is originally attributed to Kennedy, Eberhart [10] and was first intended for simulating social behaviour, as a stylized representation of the movement of organisms in a bird flock or fish school. The natural analogy of birds is that the bird flock flies in its environment looking for the best place to rest (the best place can be a combination of characteristics like space for all the flock, food access, water access or any other relevant characteristic). The basic concept of the algorithm is to create a swarm of particles which move in the space around them (the problem space or search space) searching for their goal or the place which best suits their needs given by a fitness function [11].

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor:6.887
Volume 5 Issue IX, September 2017- Available at www.ijraset.com

Fig. 1 Flock of birds

Fig. 2 School of fish

The main element of PSO is a fitness function. The specification of this function depends on the problem being optimized (especially in its dimensions) and as such it is simply referred to as $f(x_i)$ being the short for $f(x_{i,0})$,..... ,$f(x_{i,\ d})$. This function represents how well the 'i' particle's position in the multidimensional space is relatively to the desired goal. With this it weighs and binds the 'd' dimensions to be optimized, given a problem modelled as an optimization one of 'd' dimensions. Since it is multi-dimensional algorithm, the positions and velocities of the particles manipulated have 'd' components, so positions are $x_i(x_{i,0},.....,x_{i,\ d})$ and velocities as $v_i(v_{i,0}$,....., $v_{i,\ d})$.

In this algorithm there is a completely connected swarm, i.e. all the particles share information, every particle knows what is the best position ever visited by any particle in the swarm [12]. Each particle has a position (2.1) and a velocity (2.2) which are calculated as follows:

$$V_{i,d}\ (it+1)=V_{i,d}\ (it) + C1*Rnd(0,1)*[pb_{i,d}\ (it) - X_{i,d}\ (it)]+C2*Rnd(0,1)*[gb_{d}\ (it) - X_{i,d}\ (it)] \quad (2.1)$$

$$X_{i,d}\ (it+1)= X_{i,d}\ (it) + V_{i,d}\ (it+1) \quad (2.2)$$

where,

'i' is particle's index, used as a particle identifier; 'd' is dimension being considered, each particle has a position and a velocity for each dimension,

'it' is iteration number, the algorithm is iterative,

'$X_{i,\ d}$' is position of particle i in dimension d,

'$V_{i,\ d}$' is velocity of particle i in dimension d,

'C1' is acceleration constant for the cognitive component,

'Rnd' is stochastic component of the algorithm, a random value between 0 and 1,

'$pb_{i,d}$' is the location in dimension d with the best fitness of all the visited locations in that dimension of particle i,
'C2' is acceleration constant for the social component,
'gb_d' is the location in dimension d with the best fitness among all the visited locations in that dimension of all the particles.

IV. LITERATURE REVIEW

Sheta et al. [13] used Particle Swarm Optimization (PSO) to tune the parameters of the famous COnstructive COst Model (COCOMO) and also explored the advantages of Fuzzy Logic to build a set of linear models over the domain of possible software Line of Code (LOC). The performance of the developed model was evaluated using NASA software projects data set. A comparison between COCOMO tuned-PSO, Fuzzy Logic (FL), Halstead, Walston-Felix, Bailey-Basili and Doty models were provided.

Further Reddy [14] proposed Particle Swarm Optimization Algorithm (PSOA) to fine tune the fuzzy estimate for the development of software projects. Two Fuzzy software cost estimation models based on weighted average de-fuzzification were considered. The weights of the models were fine tuned using Particle Swarm Optimization Algorithm. The efficacy of the developed models were tested on 10 NASA software projects, 18 NASA projects and COCOMO 81 database on the basis of various criterions for assessment of software cost estimation models. The analysis based on VAF, Mean Absolute Relative Error, Mean Magnitude of Relative Error showed that PSOA lead to a satisfactory result.

Hari and Reddy [15] proposed two models using particle swarm optimization with constriction factor for fine tuning of parameters of the Constructive Cost estimation model (COCOMO). The model dealt efficiently with imprecise and uncertain input and enhanced the reliability of software effort estimation. The experimental part of the study illustrated the approach and contrasted it with standard numeric version of the COCOMO, standard single variable models, triangular membership functions and Gbell function models.

Hari et al. [16] introduced a new hybrid toolbox based on soft computing techniques for effort estimation. Particle swarm optimization and cluster analysis were implemented to perform efficient estimation of effort values with learning ability. The main aim of the toolbox was to provide an efficient, flexible and user friendly way of performing the effort estimation task, by catering to the needs of both the technical and the nontechnical users. The toolbox also implemented the COCOMO model to enable a comparative analysis of the proposed model. It was then observed that the model when provided with enough training data gave better results when compared with the standard COCOMO values. The PSO and K-means self learning models were employed to account for the variability in the data and as a result made efficient predictions.

Reddy et al. [17] proposed a model for software cost estimation using Multi Objective (MO) Particle Swarm Optimization. The parameters of model were tuned by using MOPSO considering two objectives: Mean Absolute Relative Error and Prediction. The COCOMO dataset was considered for testing the model. It was observed that the model gave better results when compared with the standard COCOMO model.

Reddy et al. [18] suggested that a number of models have been proposed to construct a relation between software size and Effort; however problems existed for effort estimation because of uncertainty existing in the input information. In this paper, three software effort estimation models by using soft computing technique of Particle Swarm Optimization with inertia weight for tuning effort parameters were proposed. The performance of the developed models was tested by NASA software project dataset. The developed models were able to provide good estimation capabilities.

Ziauddin et al. [19] utilized the soft computing techniques to improve the accuracy of software effort estimation. In this approach fuzzy logic was used with particle swarm optimization to estimate software development effort. The model was calibrated on 30 projects, taken from NASA dataset. Fuzzy sets were used for modeling uncertainty and imprecision in effort estimation whereas particle swarm optimization was used for tuning parameters. It was observed from the results that Fuzzy-Swarm intelligence gave accurate results as compared to COCOMO II and Alaa Sheta Model. The proposed model yielded better results in terms of MMRE.

Bardsir et al. [20] suggested that the uncertainty and complexity of software projects make the process of effort estimation difficult and ambiguous. Analogy-based estimation (ABE) is the most common method in this area because it is quite straightforward and practical, relying on comparisons between new projects and completed projects to estimate the development effort. Despite many advantages, ABE was unable to produce accurate estimates when the importance level of project features was not the same or the relationship among features was difficult to determine. In such situations, efficient feature weighting was considered as a solution to improve the performance of ABE. This paper proposed a hybrid estimation model based on a combination of a particle swarm optimization (PSO) algorithm and ABE to increase the accuracy of software development effort estimation. This combination lead to accurate identification of projects that were similar, based on optimizing the performance of the similarity function in ABE. A framework was presented in which the appropriate weights were allocated to project features so that the most accurate estimates

were achieved. The suggested model was flexible enough to be used in different datasets including categorical and non-categorical project features. Three real data sets were employed to evaluate the proposed model, and the results were compared with other estimation models. The promising results showed that a combination of PSO and ABE could significantly improve the performance of existing estimation models.

Rao et al. [21] proposed a model for tuning parameters of COCOMO model Software Cost Estimation using Multi Objective (MO) Particle Swarm Optimization. The parameters of model were tuned by using MOPSO considering two objectives Mean Absolute Relative Error and Prediction. The COCOMO dataset was considered for testing the model. It was observed that the model proposed gave better results when compared with the standard COCOMO model. On testing the performance of the model in terms of the MARE and Prediction the results were found to be useful.

Kaur and Sehra [22] proposed a model that made use of Function Points (FP), one of the size metrics used for estimating the effort of the project and Particle Swarm Optimization (PSO), a swarm intelligence technique which was used to tune the parameters of Value Adjustment Factor (VAF) thus obtaining the function count. From this optimized function count, optimized Albrecht & Gaffney effort was estimated. The estimated effort was compared with the existing effort models and performance analysis was done on the basis of %MARE (Mean Absolute Relative Error) and RMSE (Root Mean Square Error). The research showed that the results of the proposed model were far better than the existing models.

Gharehchopogh et. al. [23] proposed a new effort estimation model for software projects using Particle Swarm Optimization (PSO) and studied the effective parameters on effort estimation using the PSO algorithm. The results of the paper showed that the proposed model gave better estimation in comparison to the COCOMO model for effort.

Akhtar et al. [24] suggested that Agile Software development methodologies are gaining popularity in the software industry which can easily accommodate the requirement changes in the form of User Stories and client doesn't need to wait for long time to use the software. Researchers have used soft computing techniques to accurately estimate the effort for software development. The paper examined the effort estimation of different releases for the web based application developed using Scrum methodology and how it can be integrated with the PSO algorithm to improve the results. Team efficiency index was a crucial part of this paper which determined the actual effort of each release. Generally, the adequate target value for Mean magnitude of relative error (MMRE) was 25%. It was shown that the magnitude of relative error (MRE) for each project for the established estimation model should be less than 25% on the average. A software development effort estimation method with a smaller MMRE value gave better estimates than a model with a bigger MMRE value.

Suharjito et. al. [25] used a Neuro-fuzzy optimized model with PSO to get the right model to improve the estimation effort at NASA dataset software project. Parameter cost driver, consisting of 17 features COCOMO were then optimized using PSO techniques to get a better prediction accuracy. Furthermore, the results of the optimization were trained in using the algorithm to get a prediction Neuro-fuzzy effort. The performance of the proposed estimation model was evaluated with some other intelligent system model parameters to evaluate several criteria such as Mean Standard Error (MSE), Mean Magnitude of Relative Error (MMRE), and Level Prediction (Pred).

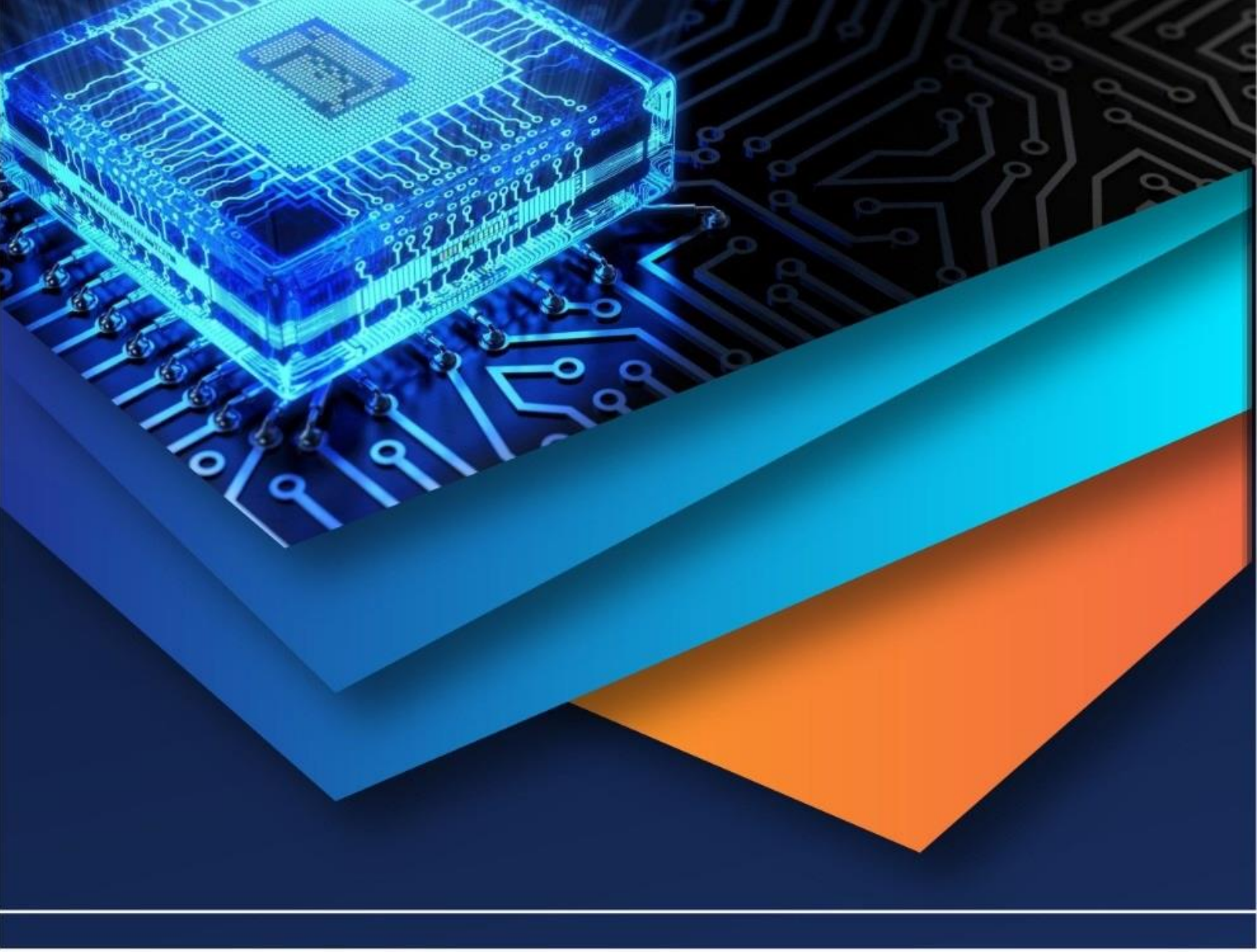
V. CONCLUSION

Software development effort estimation is a major step in software development process. It is helpful for project approval, project management, defining of project tasks and for making the team members understand their individual roles and overall goal of the project. Of late, particle swarm optimization has been widely used for optimizing the effort estimation methods. This paper presents a comprehensive survey on use of particle swarm optimization for effort estimation from the year 2008 to 2016. The literature survey suggests that there is an increased trend in use of nature inspired, computational, particle swarm optimization technique along with fuzzy logic, for optimizing the effort estimations models like COCOMO model and the results obtained are better as compared to other counterparts. This paper presents a clear view of published work in the area of effort estimation using particle swarm optimization and would further help the researchers in their future research studies and will thereby help in adding up more knowledge in this field.

REFERENCES

[1] Arshad, A., "A Critical Review of Software Size and Effort Estimation", International Journal of Computer and Electronics Research, vol 3, Issue 2, pp. 96-99, 2014.
[2] Mall, R. (2008) "Fundamentals of Software Engineering", PHI Learning Private Limited, New Delhi.
[3] Pal. G., Kumar, M., Barala, K., "A review paper on cocomo model", IJRDO - Journal of Computer Science and Engineering Vol 1, Issue-4, pp. 83-87, 2015.
[4] "IFPUG. Function Point Counting Practices Manual", Release 4.3.1.Technical report, International Function Point User Group, 2010.

[5] Sedlackova, J. "Security Factors in Effort Estimation of Software Projects", Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 3, No. 2, pp. 12-17, 2011.
[6] Albrecht, A. J. and Gaffney, J. A., "Software Function, Source Lines of Codes, and Development Effort Prediction: A Software Science Validation", IEEE Trans Software Eng. SE, vol. 9, pp. 639-648, 1983.
[7] Kemerer, C. F., "An empirical validation of software cost estimation models", Communications of ACM, vol. 30, issue 5, pp. 416-429, 1987.
[8] Matson, J. E., Barrett, B.E.; Mellichamp, J.M., "Software Development Cost Estimation Using Function Points", IEEE Transaction of Software Engineering., vol. 20, pp. 275-287, 1994.
[9] Behera, H.S. and Mishra, M., "PSO Optimized Hybridized K-Means Clustering Algorithm for High Dimensional Datasets" International Journal of Advanced Research in Computer Science, vol. 3, issue 3, pp.192-196, 2012.
[10] Bai, Q.,"Analysis of Particle Swarm Optimization Algorithm", Computer and Information Science, Vol.3, No.1, pp: 180-184, 2008.
[11] Jaganathan, P. And Jaiganesh, S., "A Particle Swarm Optimization Based Fuzzy C Means Approach for Efficient Web Document Clustering", International Journal of Engineering and Technology (IJET), vol 5, no. 6, pp. 4995- 5001, 2014.
[12] Rao, S.R. Siddaiah, P. "Design and Implementation of Four-Phase Sequences on FPGA Using Modifies Particle Swarm Optimization for Radar Applications", International Journal of Applied Engineering Research, Vol 12, Issue 11, pp. 2907-2915, 2017.
[13] Sheta, A., Rine, D. and Ayesh, A., "Development of Software Effort and Schedule Estimation Models Using Soft Computing Technique", IEEE Congress on Evolutionary Computation, Hong Kong, pp: 1283-1289, 2008.
[14] Reddy, P.V.G.D., "Particle Swarm Optimization in the fine-tuning of Fuzzy Software Cost Estimation Models", International Journal of Software Engineering (IJSE), Vol.1, No.1, pp: 12-23, 2010.
[15] Hari, CH.V.M.K. and Reddy, P.V.G.D., "A Fine Tuning Parameter for COCOMO 81 Software Effort Estimation using Particle Swarm Optimization", Journal of Software Engineering, Vol.5, No.1, pp: 38-48. 2011.
[16] Hari, CH.V.M.K., Sethi, T.S. and Jagadeesh, M., "SEEPC: A Toolbox for Software Effort Estimation using Soft Computing Techniques" International Journal of Computer Applications, Vol. 31, No. 4, pp: 12-19, 2011.
[17] Reddy P.V.G.D., Hari, CH.M.V.K. and Rao T., "Multi Objective Particle Swarm Optimization for Software Cost Estimation", International Journal of Computer Applications, Vol.32, No.3, pp: 13-17, 2011.
[18] Reddy, P.V.G.D. and Hari, CH.V.M.K., "Software Effort Estimation Using Particle Swarm Optimization with Inertia Weight" International Journal of Software Engineering (IJSE), Vol.2, No.4, pp: 87-96, 2011.
[19] Ziauddin, Tipu S.K., Zaman K., Zia S., "Software Cost Estimation Using Soft Computing Techniques", Advances in Information Technology and Management (AITM) 233 Vol. 2, No. 1, pp. 233-238, 2012.
[20] Bardsiri, V. K., Jawawi, D.N.A., Hashim, S.Z.M., Khatibi, E., "A PSO-Based Model to Increase the Accuracy of Software Development Effort Estimation", Software Quality Journal, vol. 21, issue 3, pp. 501-526, 2013.
[21] Rao, G.S., Krishna, CH. V.P., Rao, K.R., "Multi Objective Particle Swarm Optimization for Software Cost Estimation", ICT and Critical Infrstructure: Proceedings of the 48th Annual Convention of Computer Society, vol. 1, pp. 125-132, 2014.
[22] Kaur, M. And Sehra, S. K., "Particle Swarm Optimization Based Effort Estimation Using Function Point Analysis", Issues and Challenges in Intelligent Computing Techniques (ICICT), International Conference, 2014.
[23] Gharehchopogh, F.S., Maleki, I., Khaze, S.R., "A Novel Particle Swarm Optimization Approach for Software Effort Estimation", International Journal of Academic Research, Vol. 6. No. 2, pp. 69-76, 2014,
[24] Akhtar, N., Ghafir, S., Tripathi, S., " Effort Estimation of the Scrum based Software Projects using Particle Swarm Optimization", Advances in Computer Science and Information Technology (ACSIT), Vol 2, Number 7, pp. 24-26, 2015.
[25] Suharjito, Nanda S., Soewito, B. "Modeling software effort estimation using hybrid PSO-ANFIS", International Seminar on Intelligent Technology and Its Applications (ISITIA), pp. 219- 224, 2016.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY