SOFTWARE COST ESTIMATION USING FUZZY NUMBER AND PARTICLE SWARM OPTIMIZATION

Divya Kashyap[1] and D. L. Gupta[2]

[1]Computer Science and Engineering Department, IADC, Bangalore
[2]Computer Science and Engineering Department, KNIT, Sultanpur

ABSTRACT

Software cost estimation is a process to calculate effort, time and cost of a project, and assist in better decision making about the feasibility or viability of project. Accurate cost prediction is required to effectively organize the project development tasks and to make economical and strategic planning, project management. There are several known and unknown factors affect this process, so cost estimation is a very difficult process. Software size is a very important factor that impacts the process of cost estimation. Accuracy of cost estimation is directly proportional to the accuracy of the size estimation.

Failure of Software projects has always been an important area of focus for the Software Industry. Implementation phase is not the only phase for Software projects to fail, instead planning and estimation steps are the most crucial ones, which lead to their failure. More than 50% of the total projects fail which go beyond the estimated time and cost. The Standish group's CHAOS reports failure rate of 70% for the software projects. This paper presents the existing algorithms for software estimation and the relevant concepts of Fuzzy Theory and PSO. Also explains the proposed algorithm with experimental results.

KEYWORDS

Software Cost estimation, Particle swarm optimization, Fuzzy logic etc

1. INTRODUCTION

Software cost prediction is a crucial, essential and an important issue for software engineering research communities. As the software size varies from small to medium or large, the need for accuracy or correctness in software cost estimation with understanding has also grown. Software cost estimation is a method to determine effort, time and cost of a project, which in turn helps in better decision making about the feasibility and/or viability of the project. To effectively organize the project development tasks and make considerable economical and strategic planning, project management requires accurate software cost estimation. Cost estimation is a very difficult process because several known and unknown factors affect the estimation process. Software size is a very important (desirable) factor that impacts the process of cost estimation. Accuracy of cost estimation is directly proportional to the accuracy of the size estimation.

Software estimators sometimes confuse size and effort. Size, in a software development context, is the complete set of business functionalities that the end user gets when the product is deployed and is in use, and the Person months required to produce the software application of a given size is the effort.

Failure of Software projects has always been an important area of concern for the Software Industry. Implementation phase is not the only phase for Software projects to fail, instead planning and estimation steps are the most crucial ones, which leads to their failure. More than 50% of the total projects which go beyond the estimated time and cost fail.

2. MOTIVATION

Based on literature review and current trends in the software industry, some of the important motivators for this research work are discussed below:

1.) **Insignificant Contribution by the various Algorithmic Techniques**

    There are multiple studies available which accounts for the incorrect and inaccurate contribution by the various algorithmic techniques for software cost estimation for various projects. The errors are far beyond the acceptable limits and hence impacting the overall project. Since these types of traditional techniques and modeling equations based on algorithmic approach has failed, it has become imperative for us to look out for better options either in terms of new methods or optimization of the existing ones. This is quite motivating for a researcher to explore new options.

2.) **Emergence of Soft Computing & Meta-heuristic Techniques**

    Soft Computing techniques like Fuzzy systems and meta-heuristic algorithms like particle swarm optimization, firefly algorithm, harmony search, and the likes have enhanced their presence and utility in the software industry. With so many applications existing for such techniques, software cost estimation is an area which can be explored through their applications. Their advantageous characteristics can be imparted on the estimation techniques which can become more efficient. So it is also an important motivation to study the application of such techniques in the software cost estimation scenario.

3.) **Increasing criticality of Software Cost Estimation techniques & rising impact on Business Activities**

These days, the importance of Software Cost Estimation has gone beyond limits. The criticality of the software is increasing exponentially and thus cost is becoming a very important parameter. A small error in the process of cost estimation can destroy the organization’s reputation and recognition in the market. With digital world becoming so powerful, this negative publicity can spread virally. Thus cut-throat competitions are making it very difficult for everyone to survive and better techniques must be researched for cost estimation. Once the estimation process is completed successfully, one can easily execute the project without much pressure from the costing point of view.

3. INTRODUCTION TO SOFTWARE EFFORT ESTIMATION

Estimation of effort and time involved in the development of the software product under consideration are the most critical activities of the software life cycle. This process may be performed at any stage during the software development and is termed as software cost estimation. It is necessary to perform the estimate during early phases of the Software Development Life Cycle (SDCL), which helps us in taking an early decision regarding feasibility of the software development project. The results of estimation are also used for early analysis of the project efforts [1].

The primary components [2] of project development costs are: Hardware costs, Operational costs and Effort costs (in terms of Man-hours deployed on a particular project with their salaries as per their time utilized on it). The most important (dominant) cost out of all the three is the effort cost. It has been found to be very difficult to estimate the effort cost and control it, which can have a significant effect on overall costs. Software cost estimation is an ongoing process which starts at the beginning of the SDLC at the requirement gathering and proposal stage and continues throughout the conduction of a project. Projects are bound to have a specific monetary plan in terms of revenue and expenditure, and consistent cost estimation technique is necessary to ensure that spending is in line with the budget plan. Man-hours or man-months are the units used to measure the effort. An ill estimated project or poor feasibility study for a project hampers the image of the software development organization. If a project cannot be completed within the estimated time or efforts, the budget may get disturbed. This in turn can bring lots of losses for both the client and the organization. Thus we need to have a strong estimation model available with us, which can facilitate us in preparing a correct and feasible plan.

Though there has been various models proposed in the past [1] [6], but still the accuracy levels are not that satisfactory and leaves a scope of improvement over the previous work. In this paper, a new algorithm which is combination of Particle swarm optimization and fuzzy theory has been proposed and discussed for the Software cost, effort and time estimation.

4. BACKGROUND

Software cost estimation methods can be majorly categorized into three types: Algorithmic method, Expert judgment and Analogy based method [7].Machine learning is also an estimation method but is not a prominently different one in regular practices and is more categorized under the Analogy based methodology only. Each technique has its own advantages, disadvantages and limitations. With projects of large cost magnitude, many cost estimation methods should be used in parallel. This way one can compare the results produced from these methods and reliability over one particular methodology gets reduced and hence risk is also minimized [8]. But for smaller or medium sized projects, applying all the cost models might become costly affair thus making it an expensive start of the project [8].

5. FUZZY LOGIC

Fuzzy logic is a kind of multi-valued logic, which deals with approximate reasoning rather than exact and absolute reasoning. It is an approach for computing based on degree of truth rather than just true or false (1 or 0). There are few terms associated with fuzzy logic which are mentioned as below:

5.1 Fuzzy Number

A fuzzy number is a quantity whose value is uncertain rather than exact as in case of single valued numbers. Fuzzy number refers to a connected set of possible values with each value having its own weight in range 0 to 1. The weight assigned to each value is called the membership function.

5.2. Membership Function

As per the definition, for a fuzzy set A on the universe of discourse X, membership function is defined as

$$\mu A : X \rightarrow [0,1] \qquad (1)$$

where, each element of X is mapped to a value between 0 and 1[3].It represents degree of truth as an extension of valuation. Membership functions are of different shapes [9] such as triangular, trapezoidal, piecewise linear, Gaussian, bell-shaped, etc.

Trapezoidal Function: It is defined by a - lower limit, d- an upper limit, b- lower support limit, and an upper support limit c, where a < b < c < d. [3]. The Membership function depicting the Trapezoidal Function is shown below in equation 2.

$$\mu_A(x) = \begin{cases} 0, & (x < a) \text{ or } (x > d) \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d \end{cases}$$

(2)

Let A be a fuzzy number, A = [a, b, c, d; w] , w is the weight, $0 < w \le 1$ .The Trapezoidal membership of this Fuzzy number A should satisfy the following conditions:

(a) A is a continuous mapping from R to the closed interval in [0, 1].
(b) A(x) = 0, where infinite ≤ x ≤ a and d ≤ x ≤ infinite.
(c) A(x) is monotonically increasing in [a, b].
(d) A(x) = w where b≤ x≤ c.
(e) A(x) is monotonically decreasing in [c, d].

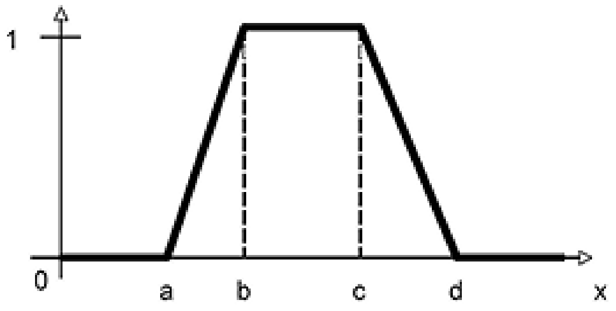

Figure 1 Trapezoidal Membership Function within the range of 0 and 1 [4]

In the Figure 1, the x axis represents the universe of discourse, whereas the degrees of membership in the [0, 1] interval is represented by y-axis [3].

6. PARTICLE SWARM OPTIMIZATION (PSO)

Particle swarm optimization (PSO) is a stochastic optimization technique invented by Dr. Eberhart and Dr. Kennedy in 1995[5], used the concept of population and social behaviour of bird flocking or schooling. PSO is an evolutionary computation technique such as Genetic algorithm. PSO optimizes problem by iteratively trying to improve a candidate (initial) solution with regard to a given measure of quality or fitness function. Initially PSO starts with candidate solutions, known as particles and move these particles around in the search-space according to the mathematical formulae describing the particle's position and velocity. Each particle's movement depends on its local best known position but, is also diverted toward the best known positions in the search space [10], [11]. These local best positions are updated as better positions are found by other particles.

6.1. PSO Algorithm

The steps of PSO Algorithm [5] are summarized as follows

```
1. For each particle i = 1, ..., S do:
   1.1 Initialize the particle's position with a uniformly distributed random
       vector: xi ~ U(blo, bup), where blo and bup are the lower and upper boundaries of the
       search-space.
   1.2 Initialize the particle's best known position to its initial position: pi ← xi
   1.3 If (f(pi) < f(g)) update the swarm's best known position: g ← pi
   1.4 Initialize the particle's velocity: vi ~ U(-|bup-blo|, |bup-blo|)
2. Until a termination criterion is met (e.g. number of iterations performed, or a solution
   with adequate objective function value is found), repeat:
   2.1 For each particle i = 1, ..., S do:
       2.1.1   Pick random numbers: rp, rg ~ U(0,1)
       2.1.2   For each dimension d = 1, ..., n do:
               2.1.2.1 Update the particle's velocity: vi,d ← ω vi,d + φp rp (pi,d-xi,d) +
                       φg rg (gd-xi,d)
       2.1.3   Update the particle's position: xi ← xi + vi
       2.1.4   If (f(xi) < f(pi)) do:
               2.1.4.1 Update the particle's best known position: pi ← xi
               2.1.4.2 If (f(pi) < f(g)) update the swarm's best known position: g ← pi
3. Now g holds the best found solution.
```

The parameters ω, φ_p, and φ_g are selected by the practitioner and control the behaviour and efficacy of the PSO method.

7. COCOMO MODEL

Developed by Barry Boehm in 1981 [12], Cost Constructive Model or COCOMO is a cost estimation model based on algorithmic properties. There were more than 60 projects which were tested and analyzed under this model before it was formally announced. Since the beginning, there were three levels of this model which were defined by Boehm. They were Basic, Intermediate and Detailed. For the purpose of this research, Intermediate COCOMO model has been used.

7.1. Intermediate COCOMO

The basic COCOMO model is based on the relationship:

$$DE = a*(SIZE)b \quad (3)$$

where SIZE is measured in thousand delivered source instructions. The constants a, b are dependent upon the ‘mode’ of development of projects, DE is Development Effort and is measured in man-months.

There were three modes proposed by Boehm- Organic, Semi-detached and Embedded Mode. Organic was associated with small teams with known development environment, Semi-detached followed a mixture of experience and fresher with mode lying between organic and embedded and finally, Embedded was used for real time projects with strict guidelines and very tight schedule. The development efforts for various modes of software in Intermediate COCOMO are calculated by equations given in table 3.1. Since the basic COCOMO was not that much accurate, intermediate COCOMO was developed with introduction of Cost Drivers.

The EAF term is the product of 15 Cost Drivers [12] that are presented below. The various multipliers of the cost drivers can be categorized from Very Low to Extra High as given.
The 15 cost drivers are broadly classified into 4 categories of product, platform, personnel and project. The meaning and classification of the cost drivers are as follows:

Table 1 Development Efforts for various modes in Intermediate COCOMO

Development Mode	Intermediate Effort Equation
Organic	$DE=EAF*3.2*(SIZE)^{1.05}$
Semi-detached	$DE=EAF*3.0*(SIZE)^{1.12}$
Embedded	$DE=EAF*2.8*(SIZE)^{1.2}$

a) Product:	RELY - Required software reliability
	DATA - Data base size
	CPLX - Product complexity
b) Platform:	TIME - Execution time
	STOR- Main storage constraint
	VIRT - Virtual machine volatility
	TURN - Computer turnaround time
c) Personnel:	ACAP - Analyst capability
	AEXP - Applications experience
	PCAP - Programmer capability
	VEXP - Virtual machine experience
	LEXP - Language experience
d) Project:	MODP - Modern programming
	TOOL - Use of software tools
	SCED - Required development schedule

With a dependency on the projects, various multipliers of the cost drivers will differ and thereby the EAF may be greater than or less than 1, thus affecting the Effort.

8. PROPOSED EFFORT ESTIMATION APPROACH

Analogy based approach [13] operates with one or two past projects selected on the basis of their similarity to the target project. Based on this approach following algorithm has been developed to estimate the effort.

ALGORITHM 1:

1. Establish attributes of planned project i.e. select all the attributes that are required for the estimation.
2. Estimate the values of attributes established in step 1.Since estimates are uncertain so fuzzy numbers are used to depict the generated values for attributes.
3. Now from the repository of historical projects find the project that closely matches the attributes of planned project.
4. Similarity distance is calculated by comparing attribute values of planned project to that of historical project. The most similar project, though closest, is still not ‘identical’ to the considered project. Hence, its ‘weight’ (as described in step 6) will be 1.

5. At this step, the values of linear coefficients may be computed using the inputs of 'similarity distances' from the existing project using PSO algorithm.
6.

The equation will be similar to linear equation, except that the weights will be adjusted in powers of distance ratios (powered to s/si, where s is the minimum distance for the best project and si for the target project).

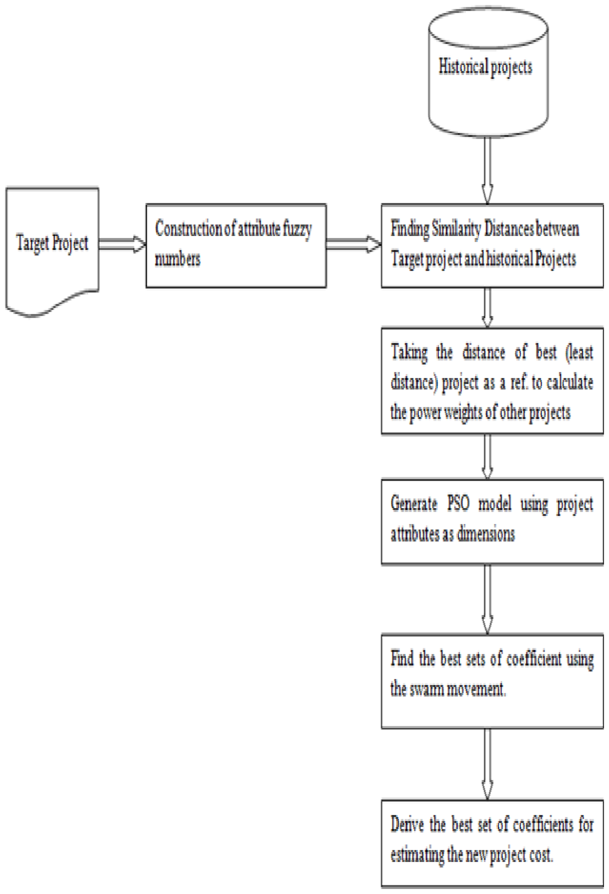
Historical projects
Target Project
Construction of attribute fuzzy numbers
Finding Similarity Distances between Target project and historical Projects
Taking the distance of best (least distance) project as a ref. to calculate the power weights of other projects
Generate PSO model using project attributes as dimensions
Find the best sets of coefficient using the swarm movement.
Derive the best set of coefficients for estimating the new project cost.

Figure 2 Generalized framework for estimation using Fuzzy numbers and PSO

The proposed model as shown in Figure 2 is considered as a form of EA (Estimation by Analogy) and comprises of following main stages:

i. Construction of fuzzy number of attributes.
ii. Finding Similarity distance between planed project and historical project.
iii. Deriving project weights according to the distance.
iv. Applying PSO algorithm to find B, as per the equation Y = B*X, where B is the coefficient vector, Y is the matrix for current project and X is the matrix for historic projects.

Construction of fuzzy number of attributes

The proposed method chooses COCOMO81 Dataset that describes the project using 15 attributes. Firstly, value of each attribute is replaced by its corresponding Fuzzy number in order to depict uncertainty. Fuzzy numbers can be constructed by any of the following two methods: [15].

a. Expert opinion
b. From data

Expert opinion is a totally subjective technique. It depends on identifying pessimistic, optimistic and most likely values for each Fuzzy number [16], whereas construction using data is based on the structure of data only. Therefore based on DATA, and using Fuzzy modelling [14], the approach develops the membership functions for each attributes. Let the numeric value of attributes in Fuzzy Number is represented in the following sequence, where trapezoidal membership has been considered as given in the section 4.

- Planned Project (P): [a1, a2, a3, a4; wa]
  and [a1, a2, a3, a4; wa] represents a fuzzy number for one attribute, where $0 \le a1 \le a2 \le a3 \le a4 \le 1$ and all are real numbers. Therefore, planned project will also have 15 sets of this type representing 15 attributes.

- Historical Project (H): [b1, b2, b3, b4; wb]
  and [b1, b2, b3, b4; wb] represents a fuzzy number for one attribute , where $0 \le b1 \le b2 \le b3 \le b4 \le 1$ and all are real numbers. Therefore, historical project will have 15 sets of this type representing 15 attributes.

Finding Similarity between planned project and historical project

To calculate the similarity distance between two fuzzy numbers, the method proposed by Azzehet. Al has been used [14] which combines the concept of geometric distance, Center of Gravity (COG) and height of generalized fuzzy numbers. The degree of similarity S(P,H) between two generalized Fuzzy numbers comprises of three elements:

a) Height of Adjustment Ratio (HAR)
b) Geometric Distance (GD)
c) Shape of Adjustment Ratio (SAF)

Equation for S(P,H) is given by the following equation[14]:

- HAR is used to assess the degree of difference in height between two generalized Fuzzy numbers. Equation for HAR is given by[4]:
  $$HAR = \sqrt{(\min(wa/wb, wb/wa))} \qquad (4)$$
  where, wa is weight for attribute of P and wb is weight for attribute of H.

- GD is used to measure the geometric distance between two generalized Fuzzy numbers including the distance between their x-axis centroid. Equation for GD is given by:
  $$GD = 1 * ((a1-b1) + (a2-b2) + (a3-b3) + (a4-b4) + (xa - xb))/5 \qquad (5)$$
- SAF is used to adjust the geometric distance, like, when the 2 Fuzzy numbers are having different shapes. Equation for SAF is given by:
- $$SAF = 1 + abs(ya-yb) \qquad (6)$$

Equation for S(P,H) is given by the following equation
$$S(P,H) = HAR*(1-GD)/SAF \qquad (7)$$

(xa, ya) and (xb, yb) presents the centre of gravity of generalized fuzzy numbers P and H respectively and calculated using equations 8 and 9:

$$ya = \begin{cases} \frac{wa \times (\frac{a3-a2}{a4-a1}+2)}{6} & , if\ a1 \neq a4 \\ \frac{wa}{2} & , \quad if\ a1 = a4 \end{cases} \tag{8}$$

$$xa = \frac{ya(a3+a2)+(a4+a1)(wa-ya)}{2wa} \tag{9}$$

Substituting the values of HAR, GD, SAF from equations 4, 5 and 6 respectively in equation 7, will give the similarity distance between one pair of attribute. Similarly, similarity distances for rest of the 14 attributes are calculated.

Now, Let Si where i varies from 1 to 15, denote similarity distance between ith attribute of the two projects we are comparing, then collectively it will give the similarity distance between two projects denoted by S, given by equation 10:

$$S = \sum_{i=1}^{15} S_i \tag{10}$$

In a similar way, similarity distance between planned project and each of historical projects are calculated to get the most similar project.

Ranking the Closest Project

After calculating the aggregated similarity measure S between the planned project and each individual analogue project, we will sort the similarity results according to their value. The project with highest similarity to target project is chosen. We choose project with least similarity distance because it has highest potential to contribute to the final estimate. We assume that the effort of the closest project is E which will be utilized for Effort adjustment.

Application of PSO algorithm for determination of Coefficients

For deriving the new estimate only using the closest project is not sufficient in many cases as [6] it may lead to bad estimation accuracy. Therefore Particle Swarm Algorithm is used to ensure that all projects are considered according to the extent to which they are similar to the test project. The framework of implementation of PSO model is shown in Figure 3. Here, the potential benefits of applying PSO to analogy-based software effort estimation models based on Fuzzy Numbers are evaluated. A suitable linear model is derived from the similarity distances between pairs of projects for obtaining the weights to be used for PSO. PSO may be considered as a process of random allocation of coefficient values, and then randomly changing them till they are closest in linking the projects (minimum error).

Figure 3, presents the iterative process to optimize the coefficients of linear equation and software cost estimation process that combines the best linear equation with estimation by analogy using Fuzzy number [15].

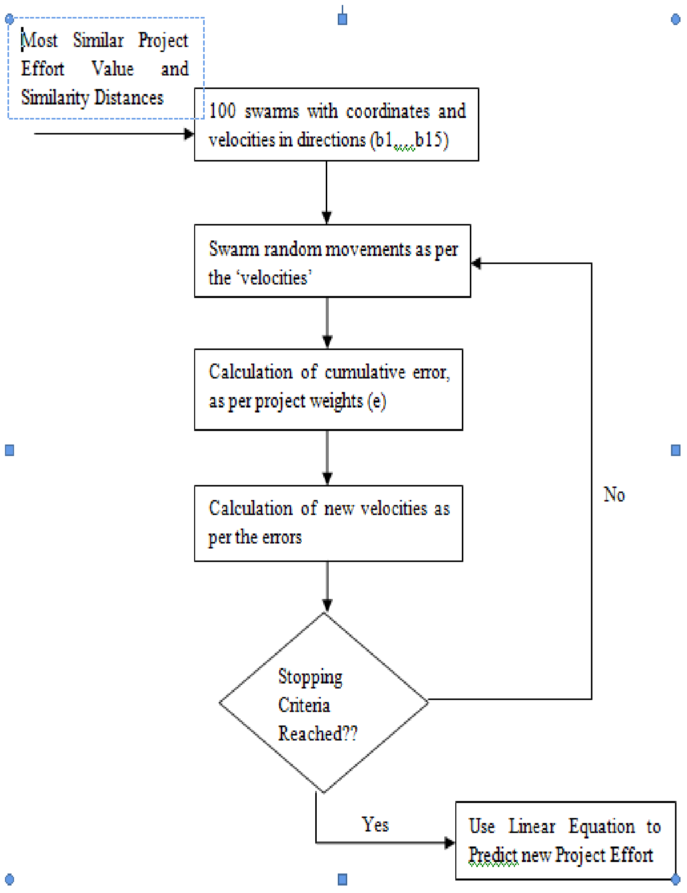
Most Similar Project Effort Value and Similarity Distances
100 swarms with coordinates and velocities in directions (b1,,,,b15)
Swarm random movements as per the 'velocities'
Calculation of cumulative error, as per project weights (e)
Calculation of new velocities as per the errors
Stopping Criteria Reached??
No
Yes
Use Linear Equation to Predict new Project Effort

Figure 3 Framework of Adjusted Fuzzy Analogy based Estimation

The algorithm for PSO implementation goes as follows:

1. Pick 100 swarms corresponding to the coordinates of 15 attributes for the equation e = Sum (abs((A(i,j)*B(j)) – E(i))^(s/si))/(no. of past projects), where A represents the values corresponding to the original scaled ratings (1-6), B is the quantity to be measured, and s is similarity ratings. Thus, dissimilar projects get lower share of contribution.
2. Pick random positions and velocities for 15 attributes for each of the swarms. Random velocity would be in the range (-x,+x), where x could take a value like 1, and position can be taken in range (0,y). Later, when equation was modified as e = Sum (abs((B(j)/A(i,j)) – E(i))^(s/si))/(no. of past projects), most of the coefficients were found in the range (0,10). The first attempt would take y = 10.
3. Compute total Error = Sum (| Estimated – Actual effort|/(Actual effort + Estimated effort)), for each project.
4. New velocity = unit random velocity * error for the swarm
5. No. of iterations may be varied as per the accuracy requirements.
6. The 'particle best' and 'global best' coordinates may be computed as per the standard PSO method.

A representation of the PSO algorithm for the project is shown in Fig. 4.

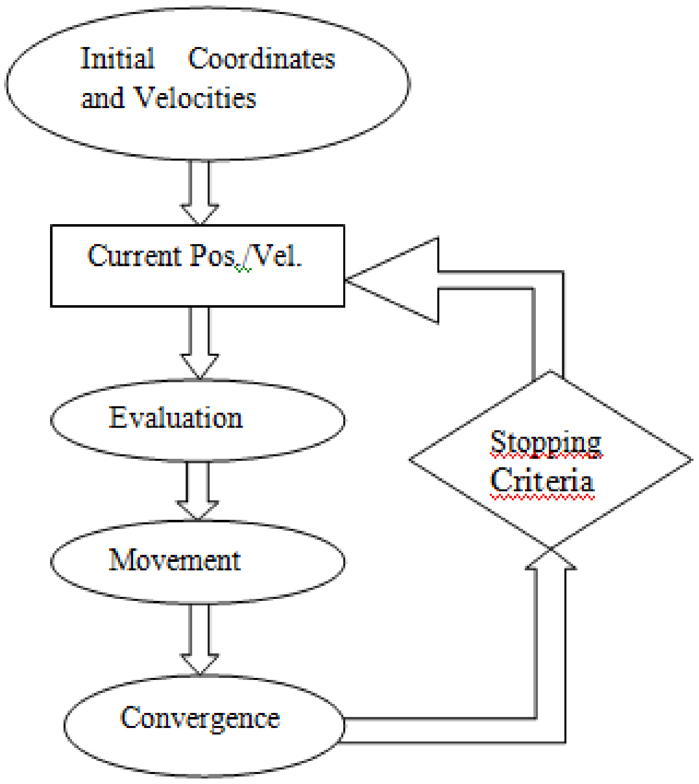

Figure 4 Adjusting Reused Effort process diagram

The failure of implementation of an iterator may come in several forms, such as

(i) Poor/Slow convergence, or slowdown before convergence near the required point,
(ii) Divergence
(iii) Undesirable oscillations
These problems may be prevented if the iteration function is mathematically consistent. This may be ensured as follows:

- The swarm movement starts slowing down when the error value start decreases, leading to a tendency to converge around the best value, as the number of iterations are increased.
- To ensure that the 15 coefficients do not go too high, the normalized data is inverted to form the equation E = Sum (B/S) + e implies e = E – sum (B/S);

  $$e\ (total) = sum\ [\ abs(\{\ E - sum(B/S)\ \})\ \hat{}\ s/si]$$

  If the absolute value of S increases, the error grows to bring it down again. This keep the values of coefficients represented by ‘S’ in a narrow range.
- An important mathematical step in the algorithm was that of ‘initialization’. If initial coefficients are taken in the range (0, 10), then there are good chances that the errors would be in the range (0, 1). If velocities are allocated in (0, 1) range, a fast convergence may be expected.

9. EXPERIMENTAL RESULTS

Using the effort of most similar project and calculated similarity distances between each attribute, the coefficients of all effort drivers in OPTIMTOOL (MATLAB (R2011a)) [104] are found that results in significantly lesser errors. There is no proof on software cost estimation models to perform consistently accurate within PRED 25% (number of observation for which MRE is less than 0.25). Applying PSO algorithm, the value of coefficients of effort drivers (shown in Table 3) is obtained that aim to minimize the MRE [204] of projects. For some of the effort drivers the coefficient value is very very less. So we can deduce that these drivers does not play important role in the adjustment of effort. For the proposed PSO algorithm, Figure 5 shows how algorithm converges with the number of iterations. Two sets of iterations per simulation set were performed in order to note the variation of precision as well. It was found that the precision also improves with the number of iterations, since the absolute difference of error of each pair of simulation decreases with the number of runs.

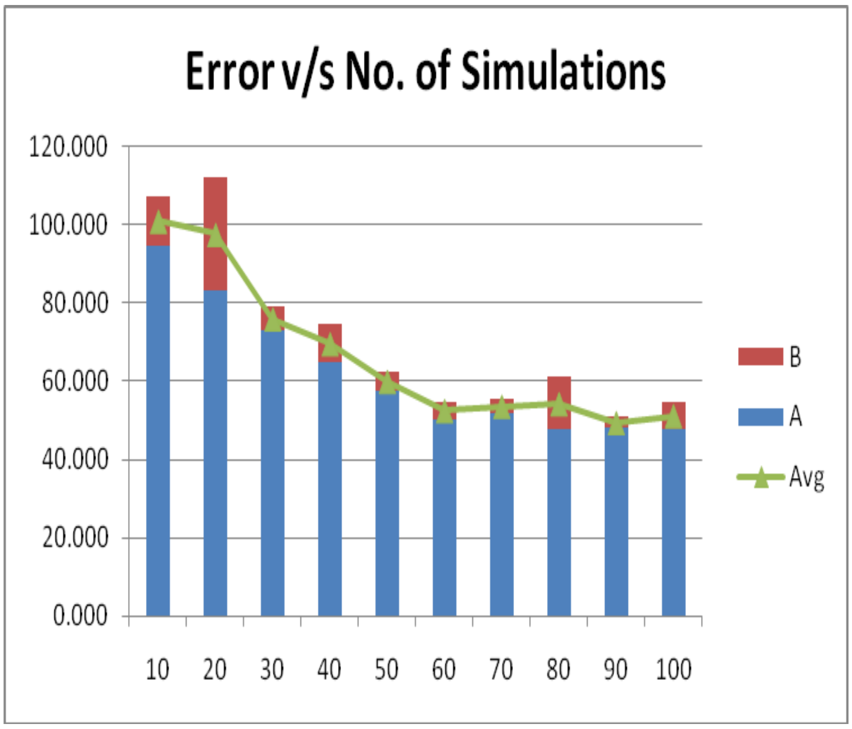

Figure 5 PSO algorithm stabilizes as the number of iterations is increased
Table 2 Corresponding to Figure 3.6

	A	B	Avg
10	94.514	12.909	100.968
20	83.157	28.896	97.605
30	73.032	5.996	76.030
40	64.894	9.600	69.694
50	57.730	4.567	60.014
60	50.379	4.414	52.586
70	51.989	3.388	53.683
80	47.565	13.645	54.387
90	48.159	2.691	49.505
100	47.819	6.980	51.309

Table 3 Best Coefficient Values Obtained through PSO

Cost Driver	**Coefficents of drivers (Bi)**
ACAP	-1.74
AEXP	4.51
CPLX	8.84
DATA	0.39
LEXP	5.43
MODP	6.33
PCAP	3.89
RELY	9.60
SCEP	1.86
STOR	0.97
TIME	-2.37
TOOL	-1.78
TURN	7.75
VEXP	0.51
VIRT	5.87

10. Conclusion And Future Work

To solve the problem of uncertain, lost values and ambiguous and vague values of attributes related to project, the concept of fuzzy numbers is used in the present proposed approach. The fuzzy model is employed in Estimation by Analogy to reduce uncertainty and improving the way to handle both numerical and categorical data in similarity measurement. Converting each attribute (real number) in to fuzzy number solve the problem of ambiguous data. Once it gets over, calculating the similarity distance of the target project with all historical projects results the most similar project. But the most similar project still has similarity distance with the project being estimated. Hence, all projects were considered in varying weights, as per their similarity distance with the proposed project – the lower the distance, the higher the weight.

Combining the concept of Fuzzy logic and PSO algorithm in a model to estimate the software effort improves the accuracy of estimation techniques. The experiments, which were conducted applying this model to NASA63 dataset, establish significant improvement under various accuracy measures. Case Based Reasoning (CBR), traditional COCOMO and estimation analogy based Fuzzy Model (EA) have been used for the comparison purpose with the proposed technique. The evaluation parameter used for the comparison is Mean Magnitude of Relative Error (MMRE).

REFERENCES

[1] Martin Shepperd, and Schofield Chris, "Estimating software project effort using analogies." Software Engineering, IEEE Transactions on 23.11, pp: 736-743, 1997.

[2] Practical Software Engineering, Accessed from http://ksi.cpsc.ucalgary.ca/courses/451-96/mildred/451/CostEffort.html on September 5, 2013.

[3] Mohammad Azzeh, Daniel Neagu, Peter I. Cowling, “Fuzzy Grey Relational Analysis for Software Effort Estimation”, Empirical Software Engineering, Vol. 15, No. 1, pp. 60-90, 2010.

[4] The fuzzy descriptive Logic System, http://gaia.isti.cnr.it/straccia/software/fuzzyDL/fuzzyDL.html

[5] R. C. Eberhart and Y. Shi, “Particle swarm optimization: Developments, applications and resources,” in Proc. IEEE International Conference on Evolutionary Computation, vol. 1, pp. 81–86, 2001.

[6] MATDOCS : http://www.mathworks.in/matlabcentral/fileexchange

[7] Fiona Walkerden, Ross Jeffery, “An empirical Study of Analogy-based Software Effort Estimation”, Empirical Software Engineering, Kluwer Academic Publishers, Boston, Volume 4, pp. 135-158, 1999.

[8] Giancarlo Attolini , “Guide to Practice Management for Small- and Medium-Sized Practices”, International Federation of Accountants ,Third Edition, 2012.

[9] Fuzzy logic membership function, http://www.bindichen.co.uk/post/AI/fuzzy-inference-membership function.html

[10] K. Vinay Kumar, V. Ravi and Mahil Carr, “Software Cost Estimation using Soft Computing Approaches”, Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques, pp. 499-518, IGI Global, 2010.

[11] Y. Shi, R. C. Eberhart , “A modified particle swarm optimizer”, Proceedings of IEEE International Conference on Evolutionary Computation, pp. 69–73, 1998.

[12] B. Boehm, “Software Engineering Economics”, Prentice Hall, 1981.

[13] Nikolaos Mittas, Marinos Athanasiades, Lefteris Angelis, “Improving Analogy-based Software Cost Estimation by a Resampling Method”, Information and Software Technology, 2007.

[14] Mohammad Azzeh, Daniel Neagu, Peter I. Cowling, “Analogy-based Software Effort Estimation Using Fuzzy Numbers”, The journal of Systems and Software 84, 2010.

[15] M. Jorgensen, “Realim in Assessment of Effort Estimation Uncertainty: It Matter How You Ask.” IEEE Transaction on software Engineering 30, 2004.

[16] Barry Boehm, Chris Abts, “Software Development Cost Estimation Approaches – A Survey”, 650 Harry Road, San Jose, CA 95120: IBM Research, pp.1-45, 1998.