Fuzzy Based Stable Maintainability Metric for Software Projects

Surender Singh[a], and Darpan an and

A
Dept. of CSE, Chandigarh University, Mohali, India
bDept. Of CSE, Chandigarh University, Mohali, India

Article History: Received: 11 January 2021; Accepted: 27 February 2021; Published online: 5 April 2021

Abstract: To make software better in view of its maintainability, its software development process must be controlled and continuously observed. Researchers and software managers have stressed on the early measurement of maintainability starting from design phase itself so that timely steps could be taken for producing maintainable software. This paper evaluates and compares several methodologies for improving the numerical stability of a fuzzy-logic-based maintainability metrics system. Fuzzy parameters are adjusted using heuristic methods. A number of alternates were considered, in which training data sets were generated using different methods and these sets were used to evaluate objective functions in GA and accordingly fuzzy parameters were tuned. After conditioning, real projects' maintainability data is used to show that fuzzy model performance is increased, however marginally, after conditioning

Keywords: Software maintainability, Fuzzy systems, Genetic algorithm.

1. Introduction

Software is described by several characteristics in its quality domain such as test-ability, maintainability, flexibility etc. The degree to which, these characteristics satisfy the requirements of software specification, expresses the quality of soft-ware. Usually a software management process has a main objective and that is to develop a software product within specified or planned cost, schedule and quality [1]. For proper control of a software development process, we need to measure software attributes at every step of software development. The measured crisp value of an attribute provides right direction to software managers to take effi-cient and effective decisions. All software attributes are not atomic; sometimes an attribute value is measured or integrated with the help of several other atomic or hybrid attributes' values. These attributes are integrated either using some mathematical formula or by using some other techniques such as fuzzy logic modeling. When contributing attributes are of diverse nature (not having same unit of measurement) and are hard to be converted into a single crisp value then it is very difficult to identify a mathematical model or formula. For such types of integra-tion, fuzzy modeling is ideal [2]. In the last decade, several fuzzy based integrated measures have come up in the literature [3] [4]. Maintainability is one such hy-brid attribute, whose crisp value depends on many lower order attributes such as source code size, comment ratio, software complexity, source code readability, documentation quality etc.

IEEE defines maintainability as "the ease with which a software system or component can be modified after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment" [5]. Software mainte-nance consumes nearly 60% to 80% [6] of software development and operations resources, so emphasis must be put on developing such software systems, whose maintenance and enhancement are relatively easy and with least possible error prone. Several researchers have stressed on the early measurement of maintainability starting from design phase itself so that timely steps could be taken for pro-ducing maintainable software [7]

Maintenance process is a summation of three main activities: program un-derstanding, impact examination and finally regression testing. Program compre-hension is activity of analyzing the source code and software artifacts in such a way to gain a complete understanding of the structure, behavior and functionality of the system being modified. Impact analysis is the activity of assessing the po-tential affect of a change with an objective of minimizing the side affects [8]. Re-gression testing is used to assess the side effects of the new changes or modifications introduced in the system as part of maintenance activities. Several models have been proposed for finding maintainability of software in different perspec-tive [9, 10, 23]. Earlier researchers used to consider only source code and its allied comments for program comprehension [11], but later with increased complexity of software, understanding other software artifacts such as design documents were also assumed compulsory [3, 20, 22]. As there are several diverse attributes collected from different artifacts of software, which are distinct in measurement and performance scale, efforts are made to integrate these, in order to get a single crisp value of maintainability. For this type of integration fuzzy modeling was used by several authors [3]. Some authors have used machine learning of en-semble techniques for estimating maintainability of software products [21].

2. Model under Consideration

For assessment four-input-boundaries practicality measurements is processed with assistance of a fuzzy model proposed by Aggarwal et. al. [3]. In these writers have considered normal average cyclomatic complexity (ACC), readability of source code (RSC), Documents quality (DOQ) and understandability of software (UOS) as significant ascribes for the estimation of viability. Practically all product that is valuable and fruitful requests changes and enhancements in the wake of being operational for quite a while relying upon the current situation. For that rea-son, maintainers need to appreciate source code and archives appropriately to comprehend the structure and conduct of programming. It has been accounted for that program perception devours the greater part of all upkeep assets. More mind boggling being the product, more troublesome it is to comprehend and hence less viable it is. McCabe's Cyclomatic Complexity [12] measures control flow complexity of software. Average Cyclomatic Complexity (ACC) is characterized as normal of Cyclomatic complexities, all things considered. Above model thinks about ACC as one of the contributing elements toward programming viability.

A source code must be upheld by enough remarks to make it intelligible with the end goal of programming cognizance. In the above model RSC is estimated utiliz-ing remark proportion in source code. DOQ measures the nature of archives with the assistance of haze's record [13]. Mist file checks the length of sentences and trouble of words. Higher the haze's list less is the nature of documentation with the end goal of understandability. To gauge the UOS, level of comparability of use of images between the language of documentation and language of source code is viewed as utilizing Laitnen's apparatus [14].

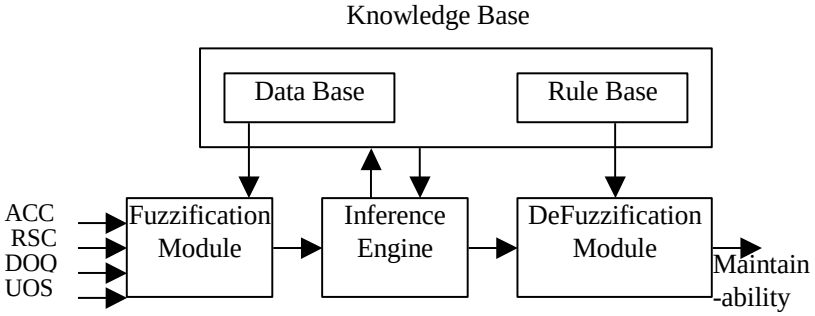


Figure 1: Maintainability Fuzzy System

The fuzzy framework, as appeared in Figure1, proposed in [3], is includ-ed four essential components: fuzzifier, fuzzy information base, fuzzy induction motor, and defuzzifier. Fuzzification module is utilized to change the fresh esti-mations of contributions to fuzzy qualities dependent on the participation capaci-ties (MFs) characterized in information base. Fuzzy information base is contained two sections: data set in which MFs are put away and rule base which stores the choice component of model. It takes care of all the lower lying modules for figur-ing the outcomes. Induction motor is the cerebrum of model. It registers the prac-ticality in fuzzy space comprises of three sub modules; specifically rule piece, suggestion and total module. Rule piece changes the fuzzified precursor some por-tion of the standard to single mathematical figure that is utilized to ensnare the yield of a standard. Total module at that point totals the individual yields of rules to a solitary fuzzy set. Defuzzifier returns the fuzzy yield of derivation module back to fresh qualities, which thus is the yield of the fuzzy framework. Fuzzy framework's presentation to a great extent relies upon the exactness of meaning of participation capacity and accuracy of rule base.

So as to fuzzify the sources of info, the accompanying MFs for the ACC, RSC, DOQ and UOS were picked by the creators [3].

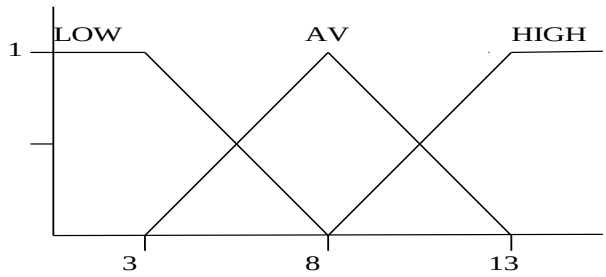


Figure 2: Input Variable ACC

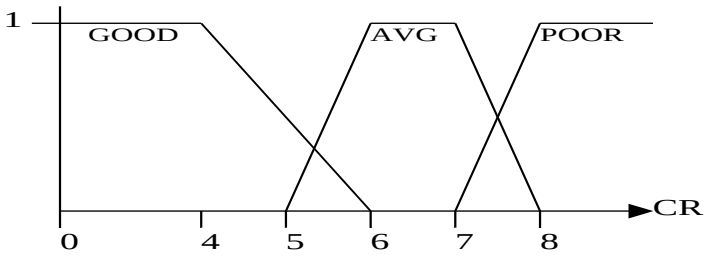


Figure 3: Input Variable RSC

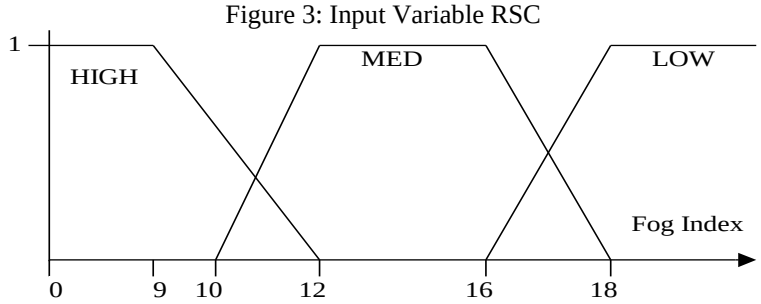


Figure 4: Input Variable DOQ

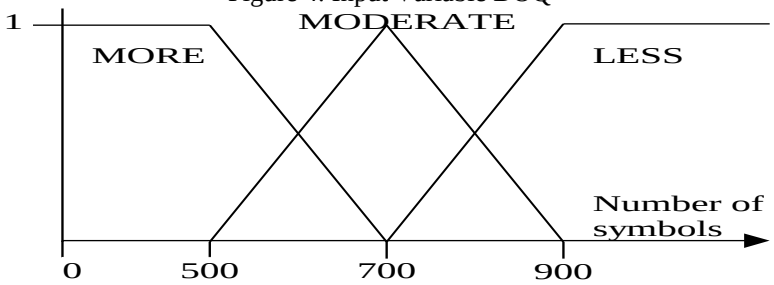


Figure 5: Input Variable UOS

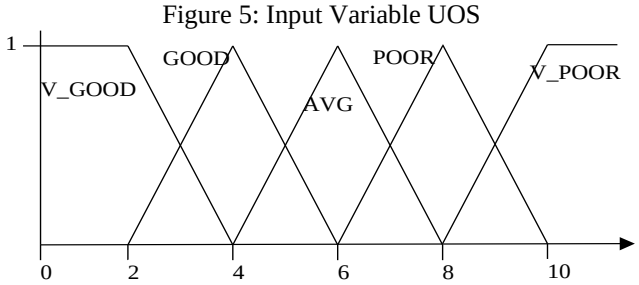


Figure 6: Output Variable Maintainability

In order to measure software maintainability using the four input metrics, the rulebase for the system consisted of the eighty-one rules as follows:

1. If (ACC is LOW) and (RSC is GOOD) and (DOQ is HIGH) and (UOS is MORE) then (MAINTAINABILITY is V_GOOD).
2. If (ACC is AV) and (RSC is GOOD) and (DOQ is HIGH) and (UOS is MORE) then (MAINTAINABILITY is V_GOOD).
3. If (ACC is HIGH) and (RSC is GOOD) and (DOQ is HIGH) and (UOS is MORE) then (MAINTAINABILITY is GOOD).

.
.
.

81. If (ACC is HIGH) and (RSC is POOR) and (DOQ is LOW) and (UOS is LESS) then (MAINTAINABILITY is V_POOR).

3. Stability Analysis of the Fuzzy System

A fuzzy model is basically an exchange work, which guides input space to yield space. A decent model, be it numerical model or heuristic model, must be steady predictable, strong and vigorous for its better presentation. A model is called mathematically steady, whenever given little Perturbation in inputs, yield doesn't differ insignificantly. This is likewise called basic steadiness [15]. Aggarwal et. al. in [4] completed affectability investigation of a fuzzy model and a neural organi-zation model of a particular picked issue. At the appropriate time, creators con-trasted the strength of the two models and the assistance of experimental proof. Strength of framework is inferred by figuring condition number for each infor-mation boundary. Condition number of each

information must be low for the en-tire framework. Condition number is characterized as the most extreme estimation of the proportion of the overall edge in the yield to the general change in in-formation over the difficult area and can be communicated as a condition as demonstrated as follows:

$$CN_x = \left| \frac{f(x + \Delta x, y, z) - f(x, y, z)}{f(x, y, z)} \times \frac{x}{\Delta x} \right| \quad (1)$$

where is condition number corresponding to input, is the fuzzy model function to measure maintainability and are the input parameters and is the perturbation in the input parameter . Perturbation must be tiny and, in our trials, we have accepted this as 0.1 percent of information boundary. The condition number is a proportion of the mathematical steadiness/molding of any model. For issues whose yield differs persistently as an element of the info, condition numbers measure "the most pessimistic scenario affectability to little bothers." [16]. Capacities with a condition number more like one are "steadier" or "very much adapted" when contrasted with capacities with a condition number more prominent than one.

Authors in [4] reasoned that neural organization-based models are more steady than fuzzy models. Yet, inaccessibility of adequate preparing information and huge preparing season of neural organization are obstruction in receiving neural organization plot for demonstrating reason. Further in the event that the frame-work for which, model is readied, is another framework, at that point we don't have any other option yet to create fuzzy model for such frameworks. According-ly, fuzzy framework models must fulfill the security standard. At the point when the framework is nonstop and boundaries are monotonic, plunges and steep-changes in the arrangement space (surface view) shows a not well molded and in-secure framework. These plunges and steep changes are estimated utilizing the idea of condition number as portrayed previously. In this way, on the off chance that we have to make framework stable, which must be, the condition number should be limited. In past paper [17], an endeavor has been made to improve the soundness of a fuzzy model-based estimation of three-inputs programming via-bility metric [2] by changing the unconditioned framework to the molded one uti-lizing hereditary calculation. The measure of molding/target work was picked as the general strength of framework, which is totaled by taking mean squared esti-mation of condition number of each info. The minimization of the characterized target work results into characterizing reexamined limits of the enrollment ele-ments of every one of the contributions viable. Our investigation inferred that framework can be made steadier to sudden changes by tuning boundaries of a framework dependent on condition number rule. This criterion has special rele-vance where previous modeling experience and/or data is unavailable. In our pre-vious methodology, we have only considered the tuning of MFs by training fuzzy model by equispaced points training data set. A training data set is made by tak-ing equispaced points from each input and then merging these in all permutations, which in turn is used in objective function. The results in previous paper were quite encouraging and prompted us to experiment with new alternatives of different training data sets and try to find out the best method for tuning MFs of fuzzy sys-tem with minimum condition number.

4. Fuzzy System Conditioning using Genetic Algorithms

Genetic Algorithms (GA) are based on Darwin's theory of 'survival of the fittest' that takes a sample of possible solutions (individuals) and employs mutation, crossover, and selection as the primary operators for optimization [18]. These are random but directed search algorithms, which have been used for optimization of difficult and discontinuous multidimensional engineering problems.

As illustrated in Figure 7, initially, GA generates a random population of individuals called chromosomes and then based on an objective function it ranks and selects individuals to build a mating pool in order to generate next generation offsprings using genetic operators such as crossover or mutation, which have the higher possibility of being fitter than the present individuals. Each individual in population is a candidate of being a solution to the problem under consideration with varying fitness values. Being a multidimensional problem, fuzzy system conditioning is also a GA application. In fuzzy system conditioning, boundaries of membership functions of already unconditioned system are varied and each individual in population defines new revised boundaries, which is also a solution to the problem, however with a varying fitness. Now each solution is checked for its fitness based on the condition criterion described in section 3.

Fuzzy framework information base has two primary segments: MFs of data sources and yields and the standard base. These segments are made utilizing master information. Some of the time, when the framework isn't an inheritance framework, there are sufficient possibilities that meanings of participation capacities and rule base are not ideal [19]. This is the primary wellspring of shakiness in the framework and same is valid for the model viable. Consequently our concern is diminished to advance the MFs and the standard base with the end goal that condition quantities of all contributions of the framework diminishes. In the current examination, we have considered advancement of the MFs just utilizing distinctive option of preparing informational collections so as to decree the best strategy for preparing so as to get a stable fuzzy model. We have done investigations in Matlab structure, in which MFs of fuzzy model depicted above are tuned utilizing GA with the end goal that general state of the framework is decreased. As there is no assurance of settling season of a condition number to a base indicated

esteem, accordingly GA is iterated for a fix number of times. In our experimentation arrangement there are 4 runs of 100 age each for one GA advancement.

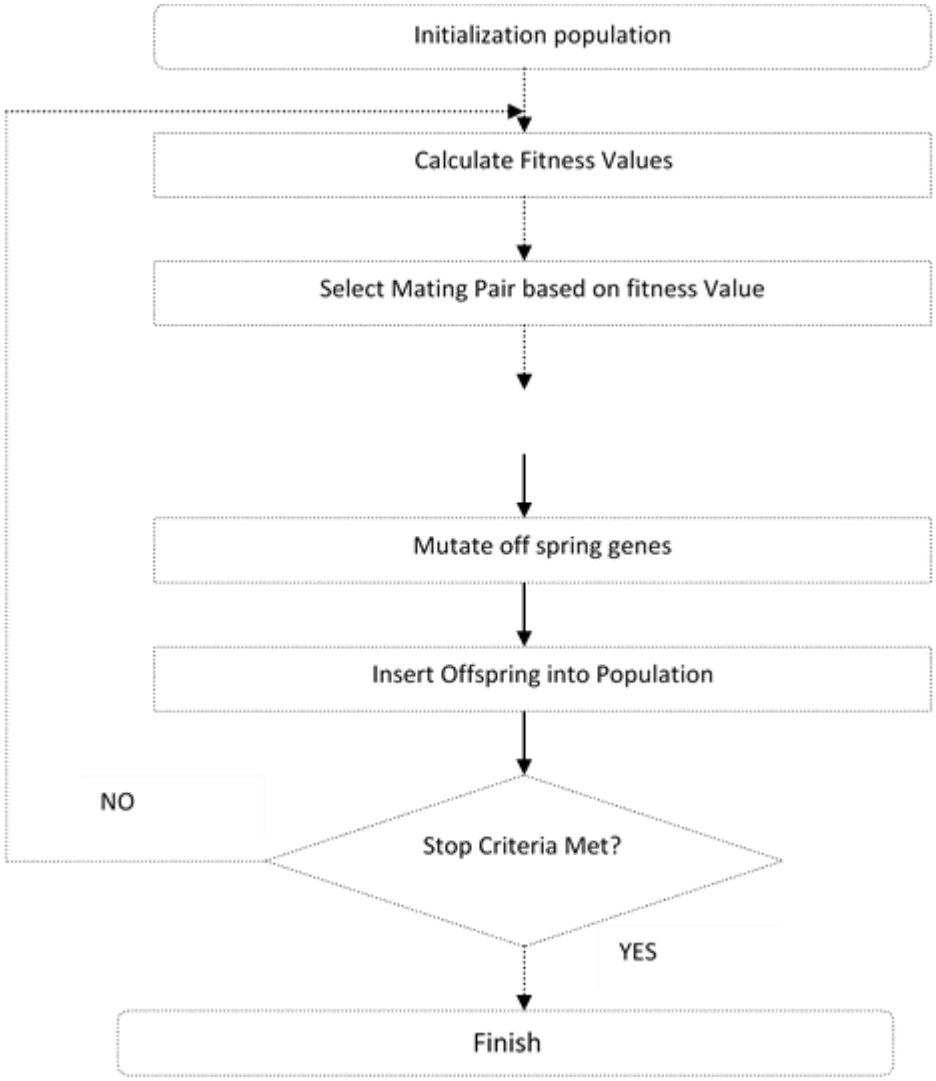
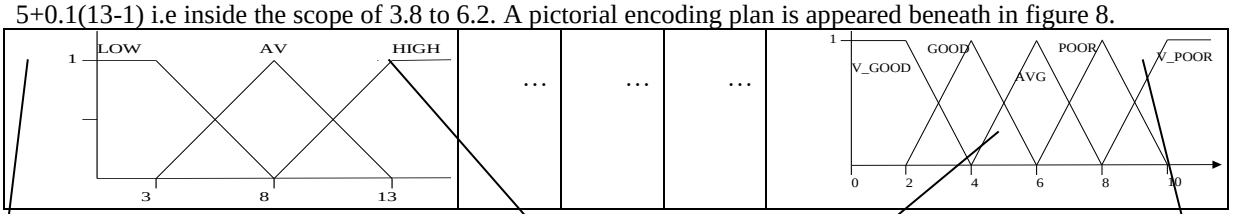


Figure 7: Major Steps of a GA algorithm

5. Genetic Encoding of Fuzzy System

In the four-data sources and single-yield framework examined over, every MF is trapezoidal (three-sided MFis a unique instance of trapezoidal MF), along these lines every MF has four boundaries, which must be balanced for enhancement. So a chromosome (plausible arrangement) will contain singular qualities as the boundaries of the apparent multitude of MFs of information sources and yields. In the model viable, there are all out 17 enrollment capacities. So there will be 17*4=68 qualities in a chromosome. So as to tune a MF, we permitted 10% variety in the current scope of every boundary. Subsequently if a boundary estimation of a MF of an information is 5 and information range is 1 to 13, at that point this boundary esteem is tuned inside the scope of 5-0.1*(13-1) and 5+0.1(13-1) i.e inside the scope of 3.8 to 6.2. A pictorial encoding plan is appeared beneath in figure 8.



MFs of Input1... MFs of Input2... MFs of Input3... MFs of Input4... MFs of output

1	2	3	4	5	6	7	8	9	10	11	12	:::::::::::::::::::::	61	62	63	64	65	66	67	68
0	1	3	8	3	8	8	13	8	13	13	15	:::::::::::::::::::::	6	8	8	10	8	10	10	12

Figure 8. Defining a chromosome

5.1 Objective function definition

This is a multi-target issue, as we have to diminish the condition number of each info. The condition number of each information can be determined utilizing condition 1 and is appeared beneath in type of condition 2-5:

$$CN_{ACC} = \left| \frac{f(ACC + \Delta ACC, RSC, DOQ, UOS) - f(ACC, RSC, DOQ, UOS)}{f(ACC, RSC, DOQ, UOS)} \times \frac{ACC}{\Delta ACC} \right|. \quad (2)$$

$$CN_{RSC} = \left| \frac{f(ACC, RSC + \Delta RSC, DOQ, UOS) - f(ACC, RSC, DOQ, UOS)}{f(ACC, RSC, DOQ, UOS)} \times \frac{RSC}{\Delta RSC} \right| \quad (3)$$

$$CN_{DOQ} = \left| \frac{f(ACC, RSC, DOQ + \Delta DOQ, UOS) - f(ACC, RSC, DOQ, UOS)}{f(ACC, RSC, DOQ, UOS)} \times \frac{DOQ}{\Delta DOQ} \right| \quad (4)$$

$$CN_{UOS} = \left| \frac{f(ACC, RSC, DOQ, UOS + \Delta UOS) - f(ACC, RSC, DOQ, UOS)}{f(ACC, RSC, DOQ, UOS)} \times \frac{UOS}{\Delta UOS} \right| \quad (5)$$

An incorporated target work is determined by taking the mean square estimation of all condition numbers, relating to every one of the information sources. This number is called condition number of the framework CNSYS and is utilized as target work for GA.

CNSYS = MSE(CNACC, CNRSC, CNDOQ, CNUOS) (6)

5.2 Generation of Training Data for MFs Tuning

In our experimentations, we have developed two modules for the purpose of genetic optimization. First is the main genetic module and is based on the algorithm described above. Second module defines an objective function which is used to calculate system condition for a given data set of inputs (here called training data set and is a matrix of 2401×4 size) based on equation 2, 3, 4, 5 and 6. Main GA module generates various acceptable solutions. For each chromosome in each population, GA module changes only membership functions' parameters from the unconditioned fuzzy model keeping rule base unchanged and then in objective function, fuzzy inference system evaluates output of system with respect to inputs provided in form of training data. Subsequently, an input is perturbed throughout in the training data set, and again same fuzzy system, which is changed by GA module, is used to evaluate outputs with respect to this perturbed data set. These two output sets are used then to calculate condition number of the system w.r.t the particular input by following equation 2 to 5. Same procedure is repeated for other inputs and then following equation 6, we get overall system conditioning for particular chromosome. In this way, all chromosomes are evaluated in objective function. We have formulated six methods for constructing training data for GA optimization as described below:

a) Conditioning with different random data set for each solution in population:

In this method a new random numbers data set of 2401 points are generated for each chromosome in population of one generation of GA algorithm for evaluating condition number of inputs and whole system. This data set is generated in the objective function each time, when this function is called by main GA module.

b) Conditioning with same random data set for each solution in population:

This method generates random numbers data set of 2401 points once in Main GA module and same is used for finding fitness for each chromosome in each population of GA algorithm. Thus, we never create training data in objective function in this method.

c) Conditioning with one input equispaced and same random numbers data set for each of rest inputs for each solution in population:

Here main function generates two data sets of 2401 points each from all inputs' ranges. One data set contains equispaced points from all inputs and another set contains random numbers from all inputs. Then both of these sets are used to find the condition numbers of all inputs. When the objective function calculates condition number for the first input then first column is taken as the equispaced vector of first input in the training set and rest three columns are taken from the random numbers generated already for another three inputs. Similar procedure is used to compute the condition number for other three inputs. So, in this method four different training data sets of 2401 points are used to calculate condition number for each input.

d) Conditioning with one input equispaced and different random numbers data set for each of rest inputs for each solution in population:

Here main GA function generates one data set which contains equispaced numbers from all inputs and another set which contains random numbers from all inputs is created by the objective function each time for each chromosome. Then these both sets are used to create four training data sets for all inputs by following the same

procedure as described in method three. Here equispaced points set remains permanent for each calculation and second data set is changed in every calculation.

e) Conditioning with all permutation generated by taking equispaced points from each input for each solution in population :

In this method seven equispaced points are taken from each input space and then these are combined in all permutations to form a training data set of 2401(7^4) points, which is used to calculate condition numbers for each input. We have taken only seven equispaced points from each input. However it can be less or more. But with the increasing size of equispaced points training set size increases exponentially and it becomes hard to complete the GA optimization within reasonable time limit. Here, this training set is created once in main module and is used for all generations and chromosomes in each population.

f) Conditioning with each input equispaced points summation for all solutions in population:

Here from each input space 2401 equispaced points are generated and then these are clubbed together to form a training data set of 2401 points. This is created in main module permanently and is used to calculate condition numbers for each input.

In order to validate the methodologies, these need to be tested with a new data set. So once the training is completed, these methods are tested with a data set of 10000 points. This data set is randomly generated for all inputs and is same for all above defined alternatives. To fully randomize the output, we repeated this process ten times and then average system condition number of all ten outputs are taken for comparison. Table 1. lists condition number of unconditioned and conditioned system for all six alternatives.

TABLE 1: Comparison of condition numbers of unconditioned and unconditioned systems

Methods	**CN_{ACC}**	**CN_{RSC}**	**CN_{DOQ}**	**CN_{UOS}**	**CN_{SYS}**
Unconditioned System	5.0622	11.945	13.314	4.4294	96.087
Conditioned System with 1st Method	2.9006	11.943	11.973	5.629	82.631
Conditioned System with 2nd Method	2.1427	4.7642	5.5817	2.2429	16.064
Conditioned System with 3rd Method	2.7784	5.8229	5.9942	3.0474	22.023
Conditioned System with 4th Method	3.6703	4.5191	6.5657	3.657	23.258
Conditioned System with 5st Method	1.895	7.6208	4.4021	8.7757	40.451
Conditioned System with 6st Method	2.384	13.577	5.8428	4.6509	62.599

Figures 9(a) to 15(a) show the graph of input UOS versus maintainability while taking constant values of other three inputs. Figure 9(a) shows Maintainability curve with respect to UOS input parameter in unconditioned System. There are steep changes and dip (marked by arrow) in the maintainability curve. Figures 9(a) to 15(a) show surface view of fuzzy model with UOS & RSC on input axes and Maintainability on output axes. Figures 9(b) shows surface view of fuzzy model with UOS & RSC on input axes and Maintainability on output axes in unconditioned system. This is also comprised of steep changes in maintainability with a gradual increase in input-parameters.

(a)UOS Vs Maintainability **(b) UOS & RSC Vs Maintainability**

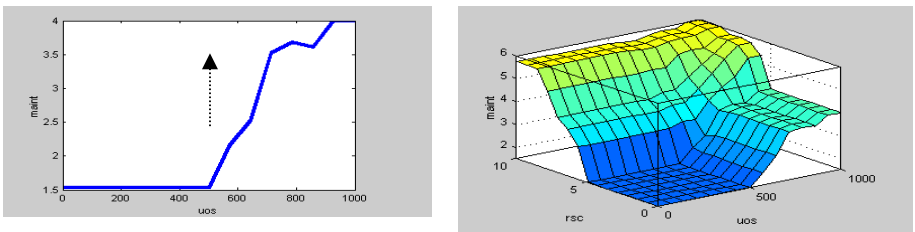


Figure 9: unconditioned System

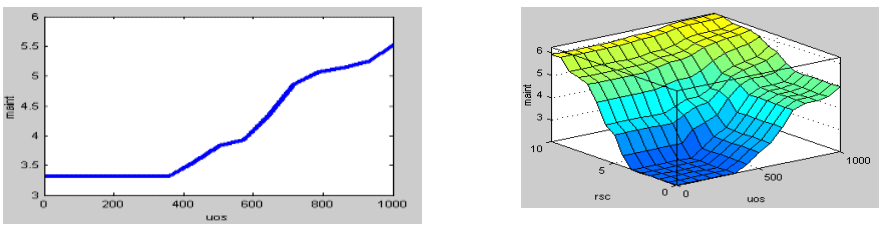


Figure 10: System is conditioned with first method

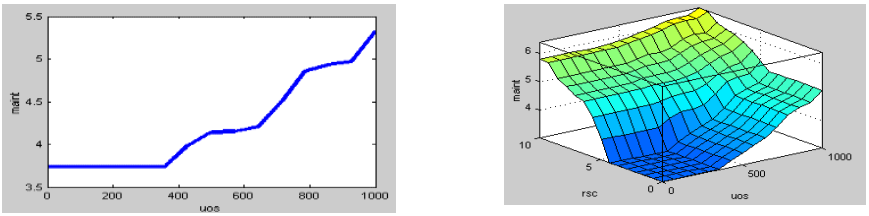


Figure 11: System is conditioned with second method

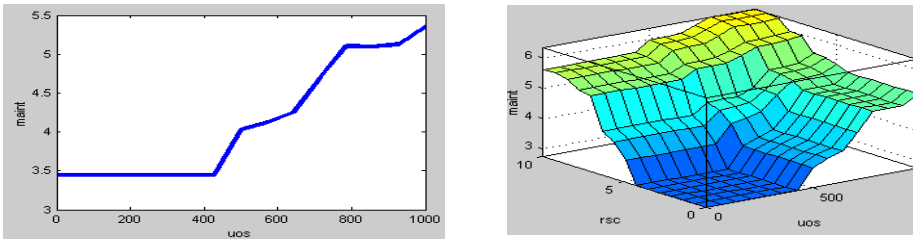


Figure 12: System is conditioned with third method

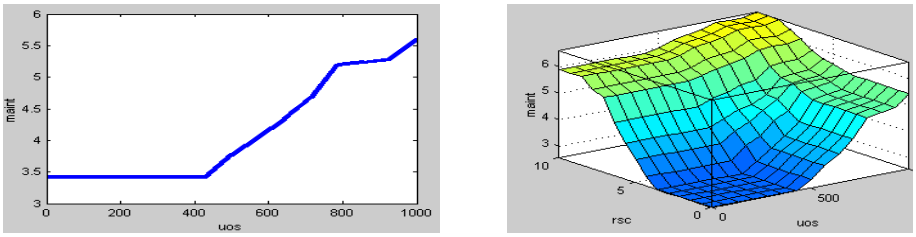


Figure 13: System is conditioned with fourth method

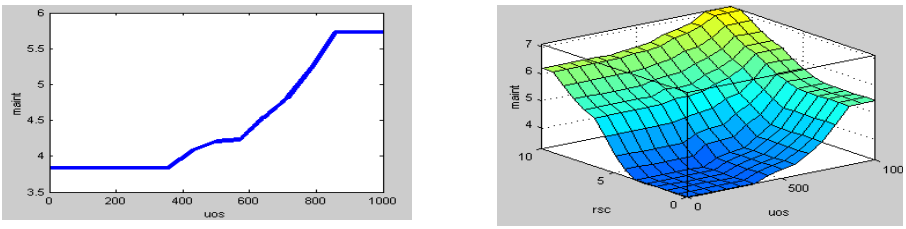


Figure 14: System is conditioned with fifth method

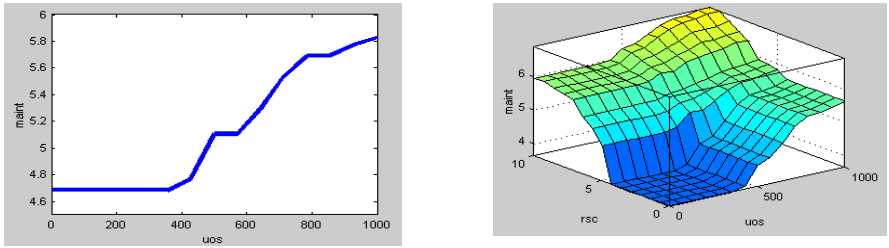


Figure 15: System is conditioned with sixth method

6. Results and Discussions

Trial results are appeared in table 1 for unconditioned and adapted frameworks. Condition number for each information boundary is diminished in every one of the six techniques. These can be decreased further on the off chance that we take a bigger preparing informational collection and condition fuzzy framework with expanded number of emphases in GA calculation.

From table, it can be deduced that Second Method is best, in which overall system condition has been reduced from 96.087 to 16.064. This is a six-fold decrease in condition number of the system. It is a significant improvement in the stability and system can be further stabilized with increased number of GA iterations. Others better methods are third and fourth methods, for which condition numbers are 22.023 and 23.258 respectively. In these three methods, which yield good conditioning, condition numbers of all the inputs as well as whole system have been less as compared to unconditioned system while on the other side first, fifth and sixth methods one or two inputs

condition number is larger than unconditioned system. Although in these methods, whole system condition number is reduced but these also destabilize system at other inputs. Figure 9(a) is of unconditioned system, in this as pointed by an arrow, there is a dip at axis (3.65, 870) which violates model consistency and continuity. In all other figures this dip has been removed completely. If we analyze the surface view of fuzzy model with UOS & RSC on input axes and Maintainability on output axes of figure 9(b), there are steep changes in maintainability given small variations in inputs for unconditioned system. These steep changes in maintainability have been replaced by smoother curves in figures 10(b) to 15(b) for conditioned system for the same surface view. If we compare the surface view of maintainability versus RSC & UOS of unconditioned system (figure 9(b)) with conditioned system of second method (Figure 11(b)) which has the least condition number, surface view is smoothest of all other methods. So from these results, we can deduce that system can be made stable or well conditioned by following second alternatives in which training data set is generated randomly and is same for each calculation throughout the GA program for the purpose of evaluation of condition number of the system. Other methods, which improve the system stability with low condition number, are method number 3 and 4.

Authors in [3] have validated their model by collecting empirical data of maintenance time of eight software projects and same has been used to evaluate the performance of the model. In this paper, we have used the same data to check the effect of conditioning on fuzzy system performance. Table 2 shows the computed maintainability values from unconditioned fuzzy model and other fuzzy model conditioned with all the six alternatives, we have discussed above, against input data from eight projects. In the last row of table 2 computed maintainability values are correlated with average maintenance time of eight projects. Here, we find that conditioning does not deteriorate the correlation in all the methodologies, rather it has been increased in each method. However, in case of best methodology, which condition the system most this increase is negligible but fourth method, which is one of the best methods for conditioning, correlation increased to 96%, which is a significant improvement. Although these methodologies are not targeted to increase correlation between maintainability and average maintenance time but this favorable change further proves the merit of our proposed methodologies for conditioning the system

Table 2: Correlation between observed maintenance time and computed maintainability values

Project Number	ACC	RSC	DOQ	UOS	Corrective maint-time	Maintainability of system when conditioned with						
						Uncon system	First Method	Second Method	Third Method	Fourth Method	Fifth Method	Sixth Method
1	8.5	3.8	11	355	11.300	3.610	3.615	3.888	3.640	3.212	3.983	4.460
2	12	7.7	15	528	21.700	7.370	6.726	6.332	6.674	6.958	7.124	6.700
3	13	5.7	11	492	18.300	5.110	5.538	5.400	5.277	5.542	5.543	5.707
4	5.4	8.3	12	567	18.000	6.810	6.027	5.311	5.546	6.116	5.966	5.692
5	15	8.9	12	363	21.100	8.000	7.492	6.750	7.331	7.275	7.409	7.051
6	7.5	7.4	8.9	390	16.100	4.560	4.826	4.948	5.127	5.290	5.152	5.763
7	11	9.2	12	451	17.900	7.070	6.758	6.473	6.438	6.375	6.942	6.543
8	9.1	6.9	13	479	17.200	6.000	6.240	6.046	6.024	5.986	6.456	6.087
Correlation between computed maintainability values and observed maintenance time →						0.873	0.902	0.875	0.913	0.961	0.899	0.912

7. Conclusion

This paper has presented six different alternatives to generate training data in order to find out the best possible method of conditioning. Subsequently each of the methods was evaluated using a new data set of 10000 points. Our initial study indicates that if training data set is created randomly from each input space and system is conditioned using this training data then this method outperforms other methods by making system more stable on average basis. However if we condition the system using training data sets generated either with one input equispaced and same random numbers data set for each of rest inputs for each solution in population or with one input equispaced and different random numbers data set for each of rest inputs for each solution in population, then these two more alternatives are also giving better results as compared to the rest three methods. These alternatives are also validated against eight real projects average maintenance time by computing maintainability from the conditioned systems and favorable change in correlation between observed maintenance time and computed maintainability values also proves the worthiness of these methodologies.

References

1. Almugrin, A.,Albattah W. , Melton, A. (2016) Using Indirect Coupling Metrics to Predict Package Maintainability and Testability The Journal of Systems & Software , doi: 10.1016/j.jss.2016.02.024.
2. Alsolai, H., Roper, M., Nassar, D. (2018) Software Maintainability in Object-Oriented Systems Using Ensemble Techniques IEEE International Conference on Software Maintenance and Evolution Predicting, DOI 10.1109/ICSME.2018.00088, 716-721.
3. D. E. Goldberg, Genetic Algorithms in search, Optimization & Machine Learning, Addison-Wesley, 1989.
4. D. Kincaid, W. Cheney, Numerical Analysis: Mathematics of Scientific Computing, 3rd Ed., Brooks/Cole – Thomson Learning Press, CA, USA, 2002.
5. Foster, J. R., "Cost Factors in Software Maintenance", Ph.D. Thesis, Computer Science Department, University of Durham, Durham, UK, 1993.
6. http://en.wikipedia.org/wiki/Structural_stability

I. Rojas, J. Gonzalez, H. Pomares, F. J. Rojas, F. J. Fernandez, A. Prieto, "Multidimensional and Multideme Genetic Algorithms for the Construction of Fuzzy System", International Journal of Approximate Reasoning, Vol. 26, 2001, Page(s): 179-210.

7. IEEE Standard Glossary of Software Engineering Terminology, Report IEEE Std 610.12-1990, IEEE, 1990.
8. J. F. Peters; W. Pedrycz, Software Engineering: An Engineering Approach, John Wiley & Sons lnc., 2000.
9. J. K. Chhabra, K.K. Aggarwal, Y. Singh, "Maintainability of Object-Oriented Software", International Journal of Management And Systems IJOMAS, 2004.
10. K. K. Aggarwal; Y. Singh; J. K. Chhabra, "A Fuzzy Model for Measure-ment of Software Maintainability & Its Performance", Int. Journal of Computer Science, USA Vol 6, No2, 2004 pg 31-43.
11. K. Laitnen, "Estimating Understandability of Software Documents", ACM SIGSOFT, Volume 21, July 1996, Page(s): 81-92.[Lamb1988] Lamb, D. A., Software Engineering: Planning for Change", Prentice Hal;l, Engineering Cliffs, NJ.1988
12. K.K. Aggarwal; Y. Singh; P. Chandra; M. Puri, "Sensitivity Analysis of Fuzzy and Neural Network Models", ACM SIGSOFT Software Engineering Notes, Vol. 30(4), July 2005 Page(s):1-4.
13. Kumar, R. & Dhanda, N. (2015). Maintainability Measurement Model for Object- Oriented Design. International Journal of Advanced Research in Computer and Communication Engineering, 4(5), 68-71.
14. Lenarduzzi, V., Sillitti, A. & Taibi, D. (2017). Analyzing Forty Years of Software Maintenance Models. IEEE/ACM 39th IEEE International Conference on Software Engineering Companion, 146-148.
15. M. Kiewkanya, N. Jindasawat, P. Muenchaisri, "A Methodology for Constructing Maintainability Model of Object-oriented Design" Proceedings of Fourth International Conference on Quality Software, QSIC, 2004 Page(s):206 – 213.
16. P. Oman; "HP-MAS: A Tool for Software Maintainability Assessment", U.I. Software Engineering Test Lab Report #92-07-ST, August 1992.
17. Pigoski, T. M., "Practical Software Maintenance – Best Practices for Managing Your Software Investment", John Wiley & Sons, New York, NY, 1997.
18. Queille, J. P., Voidrot, J. F., Wilde, N., Munro M. "The Impact Analysis Task in Software Maintenance: A Model and a Case Study", Proceedings of the International Conference on Software Maintenance, Victoria, Canada, IEEE Computer Society Press, Los Alamitos, CA, 1994, pp. 234-242.
19. R. Walker, Software Project Management: a Unified Framework, 4th Edition, Pearson Education (Singapore) Pvt. Ltd., 2004.
20. Surender Singh Dahiya, Jitender kumar Chhabra and Shakti Kumar, "Use of Genetic Algorithm for Software maintainability Metrics' Conditioning" 15th International Conference on Advanced Computing and Communications, ADCOM- 2007 (accepted for publication)
21. T. J. McCabe, "A Complexity Measure", IEEE Transactions on Software Engineering, Vol. SE-2, No. 4, Dec 1976, pp 308-319.
22. W. Pedrycz, J. F. Peters, "Computational Intelligence and Software Engineering", World Scientific, Singapore, 1998.