

An Intelligent Machine Learning Approach for Software Project Management and Performance Prediction

¹Dr. Dinesh Bhagwan Hanchate, ²Dr. Chaitany Shrikant Kulkarni

¹Professor, Computer Science and Engineering, Dnyanshree Institute Engineering and Technology, Satara
A/p Sonvadi-Gajvadi, Sajjangad Road, Tal Satara, Dist. Satara, Maharashtra, 415013, India

Email id : dbhanchate555@gmail.com

²Associate Professor, AIDS, VP'S KAMALNAYAN BAJAJ INSTITUTE OF ENGINEERING &
TECHNOLOGY, BARAMATI, Maharashtra, 413133, India.

Email id : ckulkarni47@gmail.com

Abstract.

SPM (Software Project Management) is the most important and toughest job in software engineering organizations. For successful project management, it needs the proper planning and scheduling of each software development umbrella activity. Software planning needs proper communication, requirement elicitation, suitable patterns and phase wise testing of software for fulfilling the clients requirements and completing the umbrella activities. All these can be effectively achieved and predicted by accurate estimates of the budget and schedule. Here, the literature review is illustrated by the most commonly used ML (Machine Learning) techniques such as neural net-works, case based reasoning, classification and regression trees, rule induction, genetic algorithm and genetic programming for the sub-domain of SPM. In Current era, ML imparts consistently promising accuracy in some SPM fields.

Keywords and phrases: SPM (Software Project Management), ML (Machine Learning).

How to cite this article: Dr Dinesh Bhagwan Hanchate, Dr Chaitany Shrikant Kulkarni. An Intelligent Machine Learning Approach for Software Project Management and Performance Prediction

1Dr. Dinesh Bhagwan Hanchate, 2Dr. Chaitany Shrikant Kulkarni. Int J Drug Deliv Technol. 2026;16(88s):378-385. DOI: 10.25258/ijddt.16.88s.38.

Introduction:

ML [1] always gives idea to improve the task performance by experience. ML shows its strength in various areas of applications and some interrelated domains. The training and testing of ML gives the ground truth by evaluation. ML involves learning from experience from programs which adapt their performance on a certain task over time. The paper takes a tour of the characteristics and applicability of some regularly used ML algorithms. It is difficult to make the classification of ML [3]. That's why we use to learn style and similarity function type groupings [2] The techniques and algorithms for all which are described in this section are given in the Table 1 with a classification of first paper named The Machine Learning in Software Project Management: A Journey. PART-I * published in Appl. Discrete Math Heuristic Algorithms, ISSN: 2411-7889, Samara State University, Russia-443011

RULE INDUCTION:

Rule Induction [4] is another important machine learning method. It is easier because the rules in rule induction are transparent. It is easier to interpret than a regression model or a trained neural network. This paradigm employs condition-action rules, decision trees, or similar knowledge structures. It performance element sorts instances down the branches of the decision tree or finds the first rule whose conditions match the instance, typically using an all-or none match process. Information about classes or predictions [5] is stored in the action sides of the rules or the leaves of the tree. Learning algorithms in the rule induction framework usually carry out a greedy search through the space of decision trees or rule sets, typically using a statistical evaluation function to select attributes for incorporation into the knowledge structure. Most of the methods partition the training data recursively into disjoint sets, attempting to summarize each set as a conjunction of logical conditions.

Rule Learning process

If we are given a set of training examples, i.e. instances for which classification is known we find a set of classification rules which are used to predict new cases that have not been presented to the learner before. While deriving these instances the bias imposed by languages must be taken into account such as restrictions imposed while describing data and we must also consider the language used to describe the induced set of rules. Consider a binary classification problem of classifying instances into classes of positive and negative. We are given a data description language, which impose a bias on the data, training examples, a hypothesis language imposing a bias on the

induction rules and a coverage function defining when an instance is covered by a rule. Given the above data, we need to find a hypothesis defined by a set of rules in a language, which is consistent that it does not cover any negative examples and it is complete as it covers all positive examples. In this manner, given the required data and the problem we can determine a set of rules, which classify the instances in the certain and specified problem. This forms the basis of rule induction. There are mainly two main approaches to rule induction namely relational rule learning and propositional learning.

Propositional Rule Learning

Propositional rule learning systems is suited for problems in which no substantial relationship between the values of the different attributes needs to be represented. A set of instances with known classifications where each instance is described by values of a fixed collection of attributes is given. The attributes may have either a fixed set of values or take real numbers as values. Given these instances, we then construct a set of IF-THEN rules. The output of learning is a hypothesis represented by a set of rules. After the rules have been defined, determining the accuracy of such rules and then applying these rules to practical problems analyze their quality. In propositional learning the available data has a standard form with rows being individual records or training examples and columns being properties or attributes to describe the data.

Relational Rule Learning/ Inductive logic Program-ming (ILP)

When data is stored in several tables, then it has a relational database form so such cases, the data has to be transformed into a single table in order to use standard data mining techniques. The most common data transformation approach is to select one table as the main table to be used for learning, and try to incorporate the contents of other tables by summarizing the information contained in the table into some summary attributes, added to the main table. The problem with such single-table transformations is that some information may be lost while the summarization may also introduce artifacts, possibly leading to inappropriate data mining results. What one would like to do is to leave data conceptually unchanged and rather use data mining tools that can deal with multi-relational data. ILP is intended for solving multi-relational data mining tasks. This ILP is to be used for data mining in multi-relational data mining tasks with data stored in relational databases and tasks with abundant expert knowledge of a relational nature. Another important concept within the realm of relational rule learning is that of boosting. Boosting is a particularly powerful technique and robust to enhance the prediction accuracy of

systems that learn from examples [3]. So here boosting helps to improve the overall efficiency of the results obtained.

Application Area of Rule Induction

Application areas are making Credit Decisions (in various loan companies), Diagnosis of Mechanical Devices [5], and Classification of Celestial Objects [6], Preventing breakdowns in transformers.

In [6] this paper, Pat and Herbert reviewed the major paradigms for including neural networks, machine learning, genetic learning, instance-based methods, rule induction, and analytic approaches. They considered rule induction in greater detail and reviewed some of the recent applications by how rule induction was used. Rule induction is also used by S. Kumar et al. for SPM [8]. Amrider et al [7] explained the various techniques for SCE where they also put the light on rule induction.

Benefits

Simplicity of input variables. The representation in rule-based technique is easy to depict and understand.

Limitations

No sufficient background knowledge is available and it is deduced from examples. It is hard to maintain a complex rule base.

GENETIC ALGORITHMS AND GENETIC PROGRAMMING

Genetic Algorithm (GA)

The genetic concepts of machine learning is a somewhat comparatively and relatively new concept than the other ML algorithms. The principles of biological evolution is philosophically applied by both genetic algorithms and Genetic Programming (GP) in ML and both are a form of evolutionary computing.

Mainly genetic algorithms use a vocabulary borrowed from natural genetics in that they talk about genes (or bits), population (of individuals), and chromosomes (individuals or bit strings). This approach is centralized around three main operations - crossovers, mutation and selection of individuals. Initially, many individual solutions are gathered and together to make a randomly generated population. Genetic algorithms are based upon the Darwin theory of "The survival of the Fittest". GA lies upon the fitness function. The best possible solutions are selected from the individuals pool. The brief flow of genetic algorithm is shown in Figure 24 [9]. Figure 24 shows also the Brief operational model usually used in SGA.

The best fitness individuals (Fitters) have greater chances of its selection. These have higher the probability of passing on its genetic information to future generations. After every selection is over then new individuals have to be created and these next generation individuals are formed either through crossover or mutation operations. Combining the genetic trait make up of two solution candidates forms new individuals produced a child out of two parents in crossover

operation and whereas in mutation, we alter some individuals as some randomly chosen parts of genetic information is changed to get a new individual. The generation process does not stop until one of the stop-ping criteria like minimum criteria is met or the desired fitness level is attained or a specified number of generations are reached or any combination of the above is met.

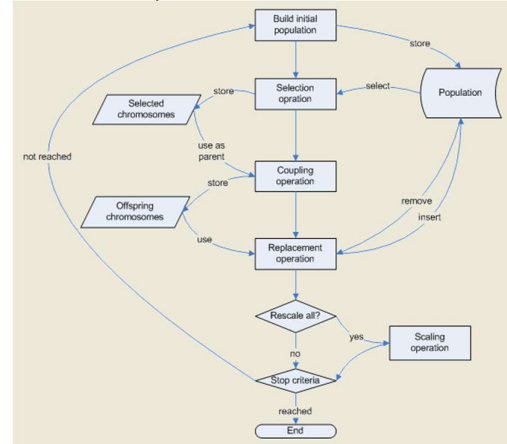


Figure 1: GA Brief operational Diagram

Genetic programming (GP)

John Koza popularized Genetic Programming [10], an offshoot of Genetic Algorithm in 1992. It is a supervised Machine Learning where algorithms are prototyped and designed after natural selection. These algorithms are represented in the form of function trees. And these trees are made to perform a given task. In GP, the fittest individuals are retained and let to develop and others are discarded. Genetic Programming works similar to genetic algorithm also obeys the principles of natural evolution to generate a solution that maximizes (or minimizes) fitness function. GP differs from GA in the way that GP gets to find the solution of a problem by representing it as an integer array while the objective of a GP is to generate a computer program to solve the optimization problem. GP cycle works like any evolutionary process. New individuals are created; tested and fitter ones succeed in creating their own children. The unfit individuals are eliminated from the population.

Originally, estimation was performed using only human expertise, but more recently attention has turned to a variety of machine learning methods. Several ML methods have been used to predict software effort, Artificial Neural Network (ANN), Case Based Reasoning (CBR), Hybrid methods, rule induction (RI). Research by Dolado shows that this technique gives promising results for effort estimation. This includes techniques which are Correlation coefficient, Adjusted mean square error-AMSE, Prediction within 25% -Pred(25), Mean Magnitude of Relative Error- MMRE, Balanced Mean Magnitude of Relative

Error BMMRE. Most commonly accuracy is defined by using MMRE, which is the mean of the absolute percentage error.

Application Area of GA

High Optimization [11],

Simulation of economic processes [12],

Scientific research purposes (Biological Evolution) and

Computer Games.

Benefits of GA

It and GA techniques can be applied to a variety of special problems.

It is based on the 'Survival of the Fittest Scheme' allowing fitter individuals to develop and discarding unfit ones.

It is easy to implement and can be easily applied without much difficulty

Limitations of GP/GA

Resource requirements are large.

It can be a time consuming process.

GA practitioners often run many copies of the same code with the same inputs to get statistically reliable results.

Rote Learning (RL)

Blessing and Watford [13] designed a scheduling system, INFMSS [14], an Intelligence flexible manufacturing system, in which they represented the knowledge by frames. Knowledge [15] was structured in three dimensions as:

FMS (flexible manufacturing system) description on one axis, Part mix on another and Schedule proposed on the third axis.

Any point if defined in the three axis space, then it was used as the best schedule. Otherwise the search for few scheduling strategies for the given part mix and FMS description was carried and simulation system was used to evaluate it. The best solution was chosen as an answer. The point where we get the best solution were saved for future references. The Figure 25 illustrates how RL work.

Figure 2: Rote learning

Limitations

Though the system saves old decisions of good results, it has no means of generalizing them. In complex environment, the chances of observing the same state are very low.

Granular Models (GM)

It is concerned with a fuzzy set-based generalization and here, we have taken as input example of the COCOMO model (f-COCOMO). The GM works as given in the Figure 26.

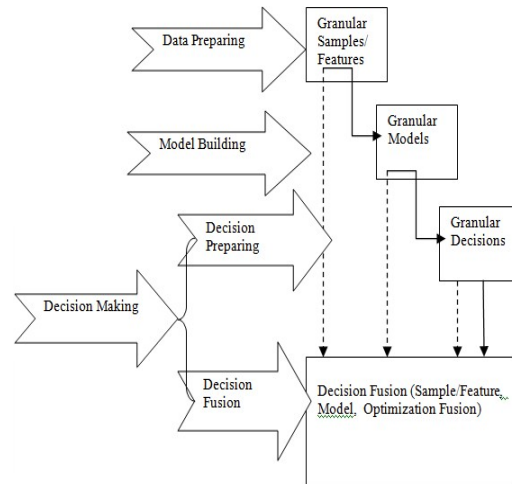


Figure 3: Granular Model and ML.

The inputs of the standard COCOMO model include an estimation of project size and an evaluation of other parameters. Rather than using a single number, the software size can be regarded as a fuzzy set (fuzzy number) yielding the cost estimate also in the form of a fuzzy set. The technology endorsed here deals with fuzzy sets. Using fuzzy sets, size of a software project can be specified by distribution of its possible values. Commonly, this form of distribution is represented in the form of a fuzzy set. For example, a small software project can be described by a fuzzy set K in the form shown in Figure 27.

Figure 4: Size of Small software project as fuzzy set

What becomes apparent here is a continuing character of belonging-ness (membership) of elements to the given concept (being here a small software project). In this sense, fuzzy sets help alleviate a dichotomy problem (yes-no evaluation) that could be very artificial to grasp using the language of set theory. In general, a fuzzy set K is described by its membership function $K(x)$ which expresses the degree of membership of x to the fuzzy set K describing a certain concept (say, small project, high reliability, etc.). Fuzzy sets (as opposed to standard interval analysis) create a more flexible, highly versatile development environment. First, they help articulate the estimates and their essence (e.g., by exploiting fuzzy numbers described by asymmetric membership functions). Second, they generate a feedback as to the resulting uncertainty (granularity) of the results.

fuzzy logic

Compared to the other classification techniques, fuzzy logic [16] provides a natural conceptual framework for representation of knowledge and inference processes based on knowledge that is imprecise, incomplete or inconsistent. The choice of the proper approach to automatic decision support depends on the problem [17]. For software classification problems, multi-branching fuzzy classing provides a more com-

prehensive solution than crisp decision trees. Such multi-branching decision support is based on structures that are not necessary trees, but also networks that resemble expert systems' structures. Especially fuzzy classification [18] proved to be effective in early identification of critical components. It must be emphasized that criticality prediction techniques being used do not attempt to detect all faults. Instead, they belong to the set of managerial instruments that try to optimize resource allocation by focusing them on areas with many faults that would affect the utility of the delivered product. The trade-off of applying complexity based predictive quality models is estimated [19] based on:

Limited resources are assigned to high-risk jobs or components;

Impact analysis and risk assessment of changes is feasible based on affected or changed complexity;

Gray-box testing strategies are applied to identified high-risk components; less customer reported failures [23] [24].

It is also worthwhile to investigate how such techniques can be further deployed to detailed design work products for even earlier defect detection [25] [26]. RBT (Requirement based testing) is the best example of combination of GA, Fuzzy, learning, requirement engineering and testing. This work is done by Ghani [20] and flow diagram is shown in Figure 28.

Figure 5: Requirement based testing SPMNet

It is the methodology which is proposed to model the development of software projects. SPMNet provides a formal modeling technique to capture the concurrent and iterative nature of software development. A tracking mechanism in the SPMNet [11] is used to keep track of all the events through the software development life cycle. Based on the previous-event information, the manager can make queries on the development history. It is an automatic technique which is based on genetic algorithm which is used to determine the resource allocation and also calculates total cost and time required for project development. Based on the information generated from genetic algorithms, the software manager will be able to predict the future states of a project by pre-executing the SPMNet. It can track the project progress, project development history and also can predict the future state of project SPMNet as a formal model to facilitate the management activities during the software life cycle. SPMNet serves as a model to assist software managers in obtaining adequate understanding on the progress of a project for themselves and explaining to others the various steps that they must go through before completing a project. In other words, SPMNet provides a communication channel between the software developers and the customers and between the

software developers themselves. It is possible to assign the probability of success rate to each decision place in the SPMNet based on the software managers experience.

Advantages of SPMNet

One of the major advantages of this model is that it provides a formal communication method between customers, software managers, and software developers. Based on this formal model, the customers and the developers will be able to work together to monitor and examine the progress of the software project.

Another advantage of SPMNet is that it provides the software developers a foundation to build tools that will support and enhance the software process.

Furthermore, a set of advanced features such as visualizing the progress of a software project, constructing a structured project plan, automatic resource allocation and scheduling, and predicting the future status of a project can be derived based on SPMNets.

Evolutionary Programming for Resource Constrained Project Scheduling

: It is dependant on the augmentation of the evolutionary program for domain-specific knowledge [21]. It goes for much load on making appropriate genetic operators for specific problem to yield a feasible solution. In this program, crossover is designed in such way to make a blind search to explore the area beyond local optima, and mutation to perform more and more intensive search to obtain an improved solution. This approach proved that it can improve the performance of evolution program in both directions of the speed and the accuracy. It can be applied and usefully for other difficult combinatorial optimization problems.

Evolution program [22] is a problem solving systems based on the principles of evolution and hereditary, such systems maintain a population of potential solutions, which undergo Darwinian selection of the survival of the fittest, and are subject to mutation and sexual recombination processes. This method lays foundations for search strategies that combine elements of genetic operators with other approaches and the problem specific information [21]. Evolution Program Application [27] to Resource Demand Optimization: During project planning, the resource demand optimization problem is classified in two ways 1. Calculate the project minimum duration, where the pre-specified resource demand level is not exceeded, 2. Optimize the resource demand level, assuming a specified (minimum) project duration. In both cases there is no method that would produce an optimal solution for the networks describing practical projects, in an acceptable time. This situation is a reason for use of heuristic algorithms, that produce satisfactory solutions, but do not guarantee obtaining the optimal solutions.

Genetic Algorithms for Resource Constrained Optimization:

There are limited solution techniques available for resource constrained project scheduling problems [28] with stochastic task durations [30]. For large deterministic problems, scheduling heuristics are found useful due to its complexity in computation. A simulation model get the performance of each combination of the heuristics taken by the genetic algorithm, and this performance information is utilized by the genetic algorithm [29] to select the next combinations to evaluate. The genetic algorithm and simulation based approach is demonstrated using a MRCPSP (Multiple Resource Constrained Project Scheduling Problem) with the stochastic task durations [31]. Simulation modeling, however, is not without drawbacks [32]. It is an expensive and time-consuming process that requires verification to make sure that it does what it is intended to do, validation to insure that it represents the real system, and sophisticated input and output analysis techniques to interpret the data. Simulation models can only give an evaluation (in terms of the performance measures) of a design that is input to the model. System design by simulation is thus an adhoc, trial and error process which is highly iterative and time consuming. This ad hoc approach is especially difficult given the complex interrelationships involved, and that several conflicting criteria must be considered during the process. Therefore, if optimization is the objective, simulation models must be used in conjunction with some optimum seeking (i.e., search) approach.

Criticality Predictions in Software Development

For cost effective software project management, one need is to focus resources on areas where they have the biggest impact. Criticality predictions are typically applied for finding out such high-impact areas in terms of most effective defect detection[33]. Criticality prediction is based on selecting a distinct small share of modules that incorporate sets of properties which would typically cause defects to be introduced during design more often than in modules that don't possess such attributes [34]. As such criticality prediction is a technique for risk analysis during the design process. Five common classification techniques for identifying critical components are: Pareto classification, classification trees, factor-based discriminant analysis, fuzzy classification, neural networks. Application in the Development Process: The criticality prediction model is used to identify critical modules for which a code inspection and later a complete module test are indispensable. The most critical of the analyzed modules are candidates for a redesign. Other high-ranking modules should have a code inspection and a high-coverage in

module test. The process for criticality classification and validation is:

Provide list of all modules at start of project, during design and after end of module test.

Provide fault history, classification for each module on the lists. A root cause analysis might be added for high ranking faults that allows for a Pareto-based mitigation list.

Provide change history, classification (i.e. number of compiles or number of deliveries).

Provide complexity classification as indicated in previous sections.

Collect and evaluate the presence of Soft Factors like new module owner.

Finalize comprehensive criticality list that takes into account the different inputs from steps 2-5 mapped to the appropriate input list.

Prepare suggestions based on ranked critical modules.

Validation and improvement of predictions based on post mortem studies with all collected faults and the population of a real criticality list. Then the actual fault ranking is compared with the predicted ranking. Reasons for deviations are investigated and used automatic classification approaches are tuned.

Among several classification techniques, best results were achieved with fuzzy classification when complexity metrics and quality data (fault rates) of several different real-time systems were given. Pareto analysis (80:20 rule) generally showed good results which clearly underline its importance as a rule of thumb for easy identification of the top 20% of critical modules. Complexity-based classification has been applied to the design and testing of telecommunication systems.

Figure 6: ML techniques three components: Representation, Evaluation and optimization Decision Tree learning (DTL)

Author et. al. did combine the decision tree learning with the input as the current problems [26]. Decision Tree can be trained with trainee example with similar type of problems, their properties and values. The DTL has helped us to classify the problem and ultimately give us the sorted problems to do analysis of the problem. Analysis and classification of the problems help us in the quality control of software products. Author et. al. has shown the defect classification as problem classification for quality control in the software project management by DTL. Author et al has also shown identification and classification of defects in software project for remedies and solutions in Software Cost Estimation [36]. Figure 23 gives DTL for defect classification and decision tree obtained in [26]

Figure 7: DTL for defect classification and Decision tree obtained.

CONCLUSION

The main aim and objective of this review is to

discuss the various Machine-Learning Techniques along with some field of SPM. The paper also gives a relatively comparison all the techniques based on their applications, advantages and limitations which can be further solved by SEI principles and SPM. We cannot conclude as any one technique being the best. Each technique has different application areas in SPM and is useful in various domains. This is dependent on their limitations. Thus, we should use that technique, keeping in mind the limitations of all the techniques. The prime focus should be on the improvement in performance and efficiency of the problem to be undertaken.

ML Applications can be in prediction and estimation transformation, generation and synthesis, in reuse, requirement acquisition and management of development knowledge [37] [38]. Jan outlined some issues in ML which relate to ambient and computational intelligence which may be applied to the SPM problems. He considered programs which are faced with the impossible tasks and concept learning in infinitely bounded time, which may be applied for the various PMM to have good learning for training an employee for quality project [39]. Figure 29 shows and gives the idea regarding the work that can be done in SPM with techniques of three components. These are Representation, Evaluation and optimization for defect detection, SCE and scheduling respectively.

References

Mitchell, M. 1999 An Introduction to Genetic Algorithms. MIT Press, NCambridge, USA, 218 p. [⇒6](#)

Przydatek, M. (2014), "Machine learning", Available at: [Iterative Dichotomiser 3 \(ID3\) algorithm.](#), [⇒6](#)

Brownlee, J. (2013), "A tour of machine learning algorithms", Machine Learning Mastery, Available at: [machinelearningmastery.com](#) [⇒6, 8](#)

Cohen, W. W. (1995), Fast effective rule induction, Proceedings of the Twelfth International Conference on Machine Learning Morgan Kaufmann Publisher pp. 115–123. [⇒6](#)

Langley, P., Simon, H. A. (1995), "Applications of machine learning and rule induction", Communications of the ACM, Vol. 38, pp. 55–64. [⇒6, 8](#)

Langley, P., Simon, H. A. (1995b), "Applications of machine learning and rule induction", Commun. ACM, Vol. 38, No. 11, pp. 54–64. [⇒8](#)

Grewal, A. S., Gupta, V., Kumar, R. (2012), "Emerging estimation techniques", International Journal of Computer Applications, Vol. 59, No. 8, pp. 30–35. [⇒8](#)

Kumar, S., Krishna, B. A., Satsangi (1994),

"Fuzzy systems and neural networks in software engineering project management", Applied Intelligence, Vol. 4, pp. 31–52. [⇒8](#)

Wall, M. (1999), "Genetic algorithm library", ., Available at: [Ge-netic algorithm library](#) [⇒9](#)

Koza, J. R. (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, USA., 680 p. [⇒10](#)

Chang, C., Christensen, M. (1999), "A net practice for software project management", IEEE Software, Vol. 16, No. 6, pp. 80–89. [⇒11, 15](#)

Hanchate, D. B., Bichkar, R. S. (2014), "Software project contacts by GRGA scheduling and EVM", International Journal of Computer Applications, Vol. 97, No. 13, pp. 1–26. [⇒11](#)

Aytug, H., Bhattacharyya, S., Koehler, G. J., Snowdon, J. L. (1994), "A Review of Machine Learning in Scheduling", IEEE Transactions on Engineering Management, Vol. 41, No. 2, pp. 165–171. [⇒11](#)

Blessing, J. A., Watford, B. A. (1987), "infms: an intelligent fms scheduling system", World Productivity Forum and 1987 Ann. Int. Industrial Engineering Con Proc., pp. 82–88. [⇒11](#)

Plunkett, K. 1993 From rote learning to system building: acquiring verb morphology in children and connectionist nets. Cognition Vol. 48, No. 1, pp. 21–69. [⇒11](#)

Garcia-Diaz, N., Lopez-Martin, C., Chavoya, A. (2013), "A comparative study of two fuzzy logic models for software development effort estimation", Procedia Technology, Vol. 7, No. 0, pp. 305–314. [⇒14](#)

Zadeh, L. A. (1994), "Soft computing and fuzzy logic.", IEEE Softw., Vol. 11, No. 6, pp. 48–56. [⇒14](#)

Dubey, S. K., Rana, A., Sharma, A. (2012), "Article: Usability evaluation of object oriented software system using fuzzy logic approach", International Journal of Computer Applications, Vol. 43, No. 19, pp. 1–6. [⇒14](#)

Jatoth, R. K., Vaddadi, M. S. B., Saladi, S. S. V. K. K. A. (2009), "An intelligent functional link artificial neural network for channel equalization", Proceedings of the 8th WSEAS International Conference on Signal Processing, Robotics and Automation (IS-PRA'09), World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, pp. 240–245. [⇒14](#)

Ghani, K., Eder, K. (2011), "Adaptive control systems for verifiably correct human-robot interaction.", University of Bristol, Available at: [Adaptive control systems for verifiably correct human-robot interaction](#) [15](#)

- Sebt, M. H., Alipouri, Y., Alipouri, Y. (2013), "Solving resource-constrained project scheduling problem with evolutionary programming", *JORS*, Vol. 64, No. 9, pp. 1327–1335. [⇒16, 17](#)
- Hart, E., Ross, P., Corne, D. (2005), "Evolutionary scheduling: A review", *Genetic Programming and Evolvable Machines*, Vol. 6, No. 2, pp. 191–220. [⇒16](#)
- Kad, S., Chopra, V. (2011), "Fuzzy logic based framework for software development effort estimation.", *An International Journal of Engineering Sciences*, Vol. 1. [⇒14](#)
- Kad, S., Chopra, V. (2012), "Software Development Effort Estimation Using Soft Computing", *International Journal of Machine Learning and Computing*, Vol. 2, No. 5, pp. 548–551. [⇒14](#)
- Kan, S. H. (2006), *Metrics and Models in Software Quality Engineering*, Low price edition, 512 p. [⇒15](#)
- Hanchate, D. B., Shabina, S., Shinde, A. S. (2010), "Defect classification as problem classification for quality control in the software project management by DTL", *IEEE International Conference on Computer Engineering and Technology (ICCET)*, Vol. 7, pp. 623–627. [⇒15, 19, 20](#)
- Kuster, J., Jannach, D., Chountas, P., Petrounias, I., Kacprzyk, J. (editors) (2008), "Extending the Resource-Constrained Project Scheduling Problem for Disruption Management", *Intelligent Techniques and Tools for Novel System Architectures Studies in Computational Intelligence*, Springer Berlin, Heidelberg Vol. 109, chapter 3, pp. 43–61. [⇒17](#)
- Kennedy, J., Rui, M. (2003), "Avoiding the pitfalls of local optima: how topologies can save the day", 6 p. [⇒17](#)
- Gonçalves, J., Mendes, J., Resende, M. (2008), "A genetic algorithm for the resource constrained multi-project scheduling problem", *European Journal of Operational Research*, Vol. 189, No. 3, pp. 1171 – 1190. [⇒17](#)
- Joshi, K., Jain, K. (2012), "A modified genetic algorithm for resource constrained project scheduling", *International Journal of Computer Applications*, Vol. 57, No. 3, pp. 41–45. [⇒17](#)
- D.B. Hanchate, Y. A. Thorat, R. H. Ambole (2012), "Review on multimode resource constrained project scheduling problem", *International Journal of Computer Science & Engineering Technology (IJCSET)*, Vol. 3, No. 5, pp. 155–159. [⇒17](#)
- Diana, S., Ganapathy, L., Pundir, A. K. (2013), "Article: An improved genetic algorithm for resource constrained project scheduling problem", *International Journal of Computer Applications*, Vol. 78, No. 9, pp. 34–39. [⇒17](#)
- Ebert, C. (1997), "Experiences with criticality predictions in software development", *SIGSOFT Softw. Engg. Notes*, Vol. 22, No. 6, pp. 278–293. [⇒18](#)
- Ebert, C., Jazayeri, M., Schauer, H. (1997), "Experiences with criticality predictions in software development", *ESEC/SIGSOFT FSE Lecture Notes in Computer Science*, Springer, Vol. 1301, pp. 278–293. [⇒18](#)
- Hanchate, D. B., Shinde, S. A., Sakhare, Y. N., Kare, S. S. (2011), "A survey of determination of effects to conflicts in project scheduling", *International journal of electronics communication and computer engineering*, Vol. 2, pp. 187–191. [⇒](#)
- Hanchate, D. B., Gawade, D. M., Shinde, A. S. (2011), "Identification and classification of defects in software project for remedies and solutions in software cost estimation", *IEEE ICCIC 2011*, Vol. 7, pp. 623–627. [⇒20](#)
- Zhang, D., Rey, J. P., Tsai, J. (2002), "Machine learning and software engineering", *Proceedings of the 14th International Conference on Tools with Artificial Intelligence*, 35 P., [⇒21](#)
- Zhang, D.-Q., Chen, S.-C. (2003), "Clustering incomplete data using kernel-based fuzzy c-means algorithm", *Neural Process. Lett.*, Vol. 18, No. 3, pp. 155–162. [⇒21](#)
- Leeuwen, V. J. 2004 *Approaches in machine learning* [⇒21](#)