ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (1052-1063), Month: April - June 2015, Available at: www.researchpublish.com

# A Novel Hybrid PSO with ABC and ANN for Software Cost Estimation

[1] S. Sunitha, [2] S. Sam Dhanasekar

[1]Master of Engineering, [2]Assistant Professor,
[1,2]Department of Computer Science and Engineering, Veerammal Engineering College Dindigul, India

***Abstract:*** **Success of project based on an accurate SCE is one of the most significant issues in project management. Based on the conducted statistical studies, it has been revealed that a majority of the software projects were not completed within the cost and time schedule on delivery time which leads to dissatisfaction of customers. Previously, a combination of Genetic Algorithm (GA) and Artificial Immune System (AIS) utilized in evaluating metrics and software criteria in SCE. The low accuracy and non-reliable structures of the algorithmic models led to high risks of software projects. So, it is needed to estimate the cost of the project annually and compare it to the other techniques. The Meta-Heuristic algorithms have been developed well lately in software fields and SCE. Meta-heuristic and Particle swarm optimization (PSO) and ABC with SVM solve the problems according to the optimization of the problems and are very efficient in optimizing the algorithmic models and the effective factors in cost estimation. In this paper we have proposed a hybrid model based on GA and ABC for optimization of the effective factors' weight in NASA dataset software projects.**

***Keywords:*** **ABC and Ann for software cost estimation, Artificial Immune System (AIS), novel hybrid PSO.**

## I. INTRODUCTION

### 1.1 INTRODUCTION TO SOFTWARE ENGINEERING:

When the first digital computers appeared in the early 1940s,[4] the instructions to make them operate were wired into the machine. Practitioners quickly realized that this design was not flexible and came up with the "stored program architecture" or von Neumann architecture. Thus the division between "hardware" and "software" began with abstraction being used to deal with the complexity of computing.

Programming languages started to appear in the 1950s and this was also another major step in abstraction. Major languages such as Fortran, ALGOL, and COBOL were released in the late 1950s to deal with scientific, algorithmic, and business problems respectively. E.W. Dijkstra wrote his seminal paper, "Go To Statement Considered Harmful", in 1968 and David Parnas introduced the key concept of modularity and information hiding in 1972 to help programmers deal with the ever increasing complexity of software systems.

The term "software engineering" was first used in 1968 as a title for the world's first conference on software engineering, sponsored and facilitated by NATO. The conference was attended by international experts on software who agreed on defining best practices for software grounded in the application of engineering. The result of the conference is a report that defines how software should be developed [i.e., software engineering foundations. The original report is publicly available.

The discipline of software engineering was created to address poor quality of software, get projects exceeding time and budget under control, and ensure that software is built systematically, rigorously, measurably, on time, on budget, and

within specification. Engineering already addresses all these issues, hence the same principles used in engineering can be applied to software. The widespread lack of best practices for software at the time was perceived as a "software crisis".

Barry W. Boehm documented several key advances to the field in his 1981 book, 'Software Engineering Economics'. These include his Constructive Cost Model (COCOMO), which relates software development effort for a program, in man-years T, to source lines of code (SLOC). T = k * (SLOC)^{(1+x)}. In 1984, the Software Engineering Institute (SEI) was established as a federally funded research and development center headquartered on the campus of Carnegie Mellon University in Pittsburgh, Pennsylvania, United States. Watts Humphrey founded the SEI Software Process Program, aimed at understanding and managing the software engineering process. His 1989 book, Managing the Software Process, asserts that the Software Development Process can and should be controlled, measured, and improved. The Process Maturity Levels introduced would become the Capability Maturity Model Integration for Development (CMMi-DEV), which has defined how the US Government evaluates the abilities of a software development team.

Software engineering is concerned with all as pects of software production from the early stages of system specification through to maintaining the system after it has gone into use.

**Defining Cost Estimation:**

Cost estimation can be defined as the approximate judgement of the costs for a project. Cost estimation will never be an exact science because there are too many variables involved in the calculation for a cost estimate, such as human, technical, environmental, and political. Futhermore, any process that involves a significant human factor can never be exact because humans are far too complex to be entirely predictable. Furthermore, software development for any fair-sized project will inevitably include a number of tasks that have complexities that are difficult to judge because of the complexity of software systems.

Cost estimation is usually measured in terms of effort. The most common metric used is person months or years (or man months or years). The effort is the amount of time for one person to work for a certain period of time. It is important that the specific characteristics of the development environment are taking into account when comparing the effort of two or more projects because no two development environments are the same. A clear example of differences in development environments are the amount of time people work in different countries; the typical workweek in North America is 40 hours per week, while in Europe the typical workweek is 35 hours per week (Londeix, 1987). Thus, when comparing a project from North America with a project from Europe, a conversion factor would have to be used to all for an accurate comparison. Different variables can be used for cost estimation, which leads to a difficulty when comparing projects if standard models or tools are not used. For example, a cost estimate can include factors from management, development (e.g., training, quality assurance), and other areas specific to an organization.

**Cost Estimation and Project Planning:**

Cost estimation is an important tool that can affect the planning and budgeting of a project. Because there are a finite number of resources for a project, all of the features of a requirements document can often not all be included in the final product. A cost estimate done at the beginning of a project will help determine which features can be included within the resource constraints of the project (e.g., time). Requirements can be prioritized to ensure that the most important features are included in the product. The risk of a project is reduced when the most important features are included at the beginning because the complexity of a project increases with its size, which means there is more opportunity for mistakes as development progresses. Thus, cost estimation can have a big impact on the life cycle and schedule for a project.

Cost estimation can also have an important effect on resource allocation. It is prudent for a company to allocate better resources, such as more experienced personnel, to costly projects. Manpower loading is a term used to measure the number of engineering and management personnel allocated to a project in a given amount of time. Most of time, it is worse for a company if a costly project fails than if a less costly project fails. When tools are used for estimation, management and developers can even experiment with trading off some resources (or factors) with others while keeping the cost of the project constant. For example, one tradeoff may be to invest in a more powerful integrated development environment (IDE) so that the number of personnel working on a project could be reduced. Cost estimation has a large impact on project planning and management.

**Cost Estimation During the Software Life Cycle:**

Cost estimation should be done throughout the entire life cycle. The first time cost estimation can be done is at the beginning of the project after the requirements have been outlined. Cost estimation may even be done more than once at the beginning of the project. For example, several companies may bid on a contract based on some preliminary or initial requirements, and then once a company wins the bid, a second round of estimation could be done with more refined and detailed requirements. Doing cost estimation during the entire life cycle allows for the refinement of the estimate because there is more data available. Periodic re-estimation is a way to gauge the progress of the project and whether deadlines will be able to be met.

Effective monitoring and control of the software costs is required for the verification and improvement of the accuracy of the estimates. Tools are available to help organize and manage the cost estimates and the data that is captured during the development process. People are less likely to gather data if the process is cumbersome or tedious, and so using tools that are efficient and easy to use will save time. It is not always the most expensive tool that will be the best tool to buy, but rather the tool that is most suited to the development environment. Some thought should be given to the level of detail at which the metrics will be gathered, as well as planning for what metrics may be used in the future for comparison with other projects. The metrics that are gathered will be highly dependent upon the organization's development and organizational practices.

The success of a cost estimate method is not necessarily the accuracy of the initial estimates, but rather the rate at which the estimates converge to the actual cost. An organization that does a great deal of contract work would place more importance on the initial estimates. However, in general, the method will be better if it converges quickly to the actual cost of the project. At the end of the project, all estimation methods have the opportunity to converge to the actual cost because enough information is available.

**The Estimator:**

The people who do the cost estimates could be either directly or indirectly responsible for the implementation for a project, such as a developer or manager, respectively. Someone who has knowledge of the organization and previous projects could use an analogy-based approach to compare the current project with previous projects, which is a common method of estimation for small organizations and small projects. The historical data is often limited to the memory of the estimator. In this case, the estimator would need to be experienced and would likely have been with the company for a while.

Some people believe it is better if the estimates are done by outsiders so that there is less chance of bias. It is true that people outside an organization will likely have to deal with fewer company politics than people within the organization. For example, the developer for a company may want to please the manager and so give an estimate that is overly-optimistic. The disadvantage of having an outside estimate is that the person would have less knowledge of the development environment, especially if the person is from outside the company. An empirical method of estimation would then be required, such as the Constructive Cost Model (COCOMO). Empirical methods of estimation can be used by all types of estimators. There may be some resistance to using an empirical method of estimation because there may be some question on whether a model could outperform an expert. People who are accurate estimators are rare in our experience, and so it is best to get the opinion of several people or tools.

**General Steps and Remarks:**

To give the reader a better idea of how software cost estimation fits into the development process, we will outline the general steps for doing cost estimation. The steps are not numbered because they are not completely discrete from one another. As well, although they generally follow a logical order, some of the steps can fit into several parts of the development process. Although this may at first seem to be confusing, the steps are straightforward enough that there should not be any difficulty in envisioning how they fit into the development process.

The first and most important step is to establish a cost estimate plan (Pressman, 2001). In this plan, it should be stated what data will be gathered, why the data is being gathered, and the goal for doing the cost estimation process. Determining which data is to be gathered is essentially stating the level of detail of the metrics. This decision can

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (1052-1063), Month: April - June 2015, Available at: www.researchpublish.com

influence the amount of decomposition for the tasks. There is obviously no point in gathering data that will not be used. This will seem unnecessary, and require more work, for the people who have to collect and manage the data. Although it may seem like a good idea to gather metrics that will not be used in the near future, but could possibly be used in the future, this is a waste of resources at the time. A fair amount of thought should be put into the cost estimation plan, much like the requirements for a project.

The second step is to perform a cost estimation based on the requirements. Decomposition of the project can be done at this time if a lower level of abstraction is needed for the data. Keep in mind that it is important to use more than one method of estimation because there is no perfect technique. If there are wide variances in the estimates of the methods, then the information used to make the estimates should be re-evaluated (Humphrey, 1990).

During the lifecycle, re-estimates should be done to allow for refinement of the cost estimates. The re-estimates could be done at major milestones during the project, or at specific time intervals. This decision will depend on the situation. Changes may have to be made to the project if the cost estimates either increase or decrease.

At the end of the project, a final assessment of the results of the entire cost estimation process should be done. This allows a company to refine the estimation process in the future because of the data points that were obtained, and also allows the developers to review the development process.

**Cost Estimation Process:**

In order to understand the end result or the outputs of the software cost estimation process we must first understand what software cost estimation process is. By definition, software cost estimation process is a set of techniques and procedures that is used to derive the software cost estimate. There is usually a set of inputs to the process and then the process uses these inputs to generate or calculate a set of outputs.

**Classical View:**

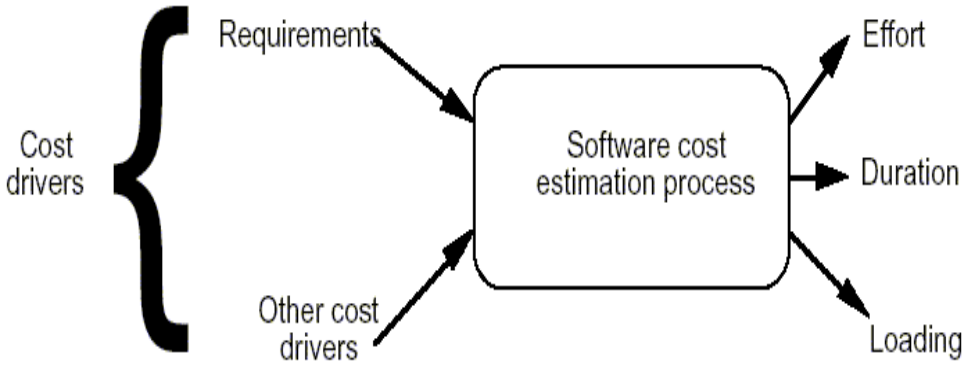



**Figure 1: Classical view of software estimation process (Vigder and Kark, 1994)**

Most of the software cost estimation models views the estimation process as being a function that is computed from a set of cost drivers. And in most cost estimation techniques the primary cost driver or the most important cost driver is believed to be the software requirements. As illustrated in figure 1, in a classical view of software estimation process, the software requirements are the primary input to the process and also form the basis for the cost estimation. The cost estimate will then be adjusted accordingly to a number of other cost drivers to arrive at the final estimate. So what is cost driver? Cost driver is anything that may or will affect the cost of the software. Cost driver are things such as design methodology, skill-levels, risk assessment, personnel experience, programming language or system complexity.

In a classical view of the estimation process, it will generate three outputs - efforts, duration and loading. The following is a brief description of the outputs: Manpower loading - number of personnel (which also includes management personnel) that are allocated to the project as a function of time. Project duration - time that is needed to complete the project. Effort - amount of effort required to complete the project and is usually measured in units as man-months (MM) or person-months (PM).

The outputs (loading, duration and effort) are usually computed as fixed number with or without tolerance in the classical view. But in reality, the cost estimation process is more complex than what is shown in figure 1. Many of the data that are inputs to the process are modified or refined during the software cost estimation process.

**Actual View:**

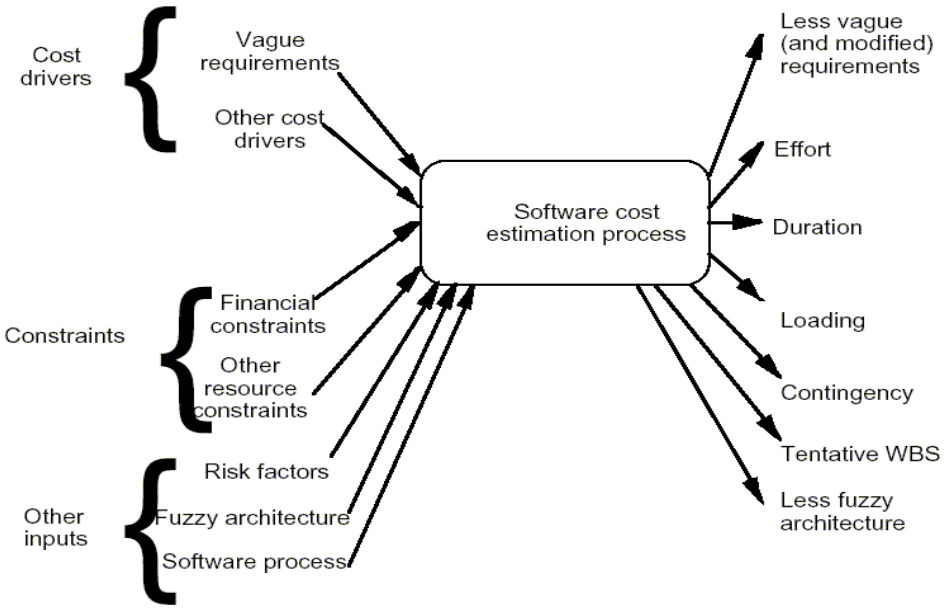



**Figure 2: Actual Cost Estimation Process (Vigder and Kark, 1994)**

In the actual cost estimation process there are other inputs and constraints that needed to be considered besides the cost drivers. One of the primary constraints of the software cost estimate is the financial constraint, which are the amount of the money that can be budgeted or allocated to the project. There are other constraints such as manpower constraints, and date constraints. Other input such as architecture, which defines the components that made up the system and the interrelationships between these components. Some company will have certain software process or an existing architecture in place; hence for these companies the software cost estimation must base their estimates on these criteria.

There are only very few cases where the software requirements stay fixed. Hence, how do we deal with software requirement changes, ambiguities or inconsistencies? During the estimation process, an experienced estimator will detect the ambiguities and inconsistency in the requirements. As part of the estimation process, the estimator will try to solve all these ambiguities by modifying the requirements. If the ambiguities or inconsistent requirements stay unsolved, which will correspondingly affect the estimation accuracy.

**Cost Estimation Accuracy:**

The cost estimation accuracy helps to determine how well or how accurate our estimation is when using a particular model or technique. We can assess the performance of the software estimation technique by:

- Absolute Error (Epred - Eact)
- Percentage or Relative Error (Epred - Eact) / Eact
- Mean Magnitude of Relative Error

Each of the error calculation techniques has advantages and disadvantages. For example, absolute error fails to measure the size of the project, and mean magnitude of relative error will mask any systematic bias (don't know if the estimation is over or under).

louds should not know the query but should be able to return the records that satisfy the query. This is achieved by means of searchable encryption.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (1052-1063), Month: April - June 2015, Available at: www.researchpublish.com

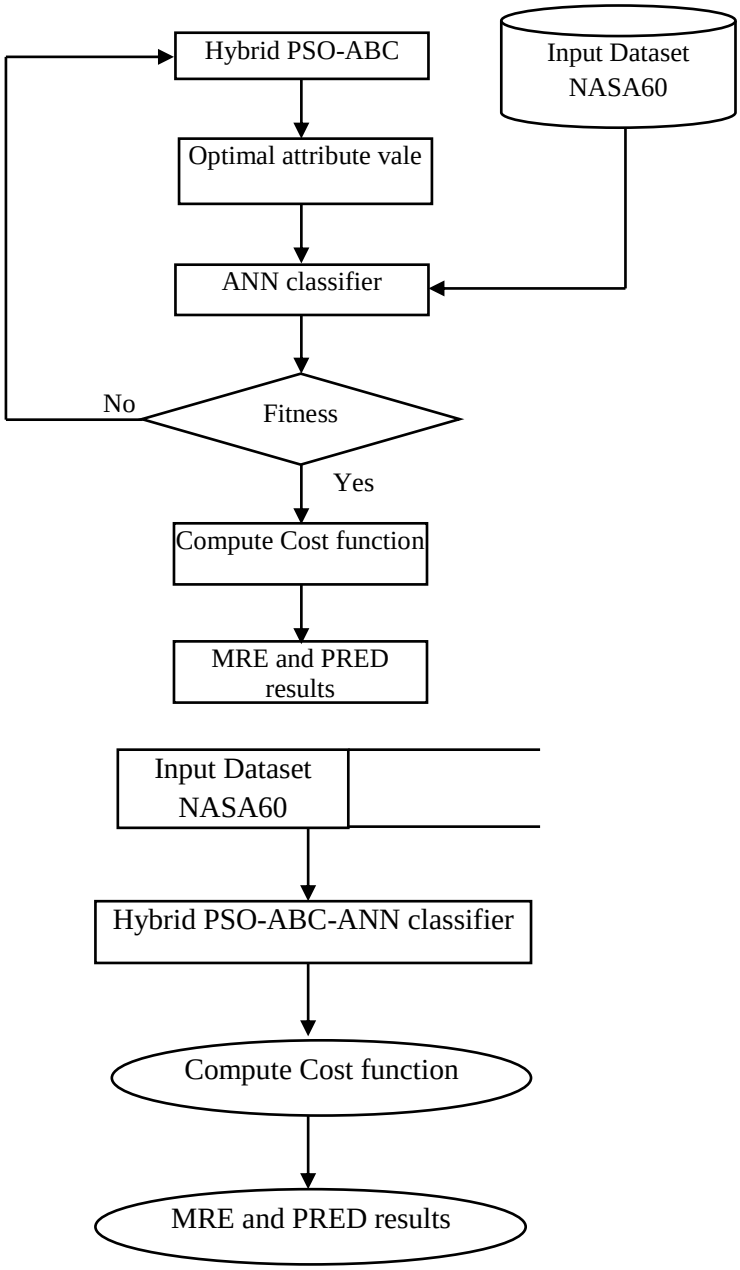
### 1.2 NEED:

In software project issues improper estimation of timing and cost at the beginning of deigning phase causes the management and planning of the project to be based on irrational estimations and thus the obtained estimation is unreal and cannot be relied on. On the other hand, wrong estimations bring about false expectations in customers and **also** imposes financial and time loses on company.

### 1.3 OBJECTIVE:

Estimation based on software experts comments: it shows the rational and mental process of effort estimation and is often based of prior experiences in developing and managing similar projects. Algorithmic estimation: this kind of estimation is used in estimation and production of projects which obviously shows the relationship between the efforts of one or several project properties using linear equations. These techniques have been the most prevalent ones in SCE so far.

## II. ARCHITECTURES



ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (1052-1063), Month: April - June 2015, Available at: www.researchpublish.com

## III. METHODOLOGY

Planning and determining the cost is one of the most important activities in software projects to produce and develop software projects. SCE is considered one of the most difficult issues of management in software projects. Effort Multipliers (EM) factors determine the required effort to complete the software project and SCE factors in NASA60 dataset projects include programmer development tools data set size and etc. [33]. Development team cost, testing and implementation, software management cost and control management cost consist the most important project cost which all requires acceptable result and estimation with an error value close to the real value [33]. SCE deals with estimation of probable cost and time for project completion. In general SCE is based on anticipation of LOC size. The obtained estimation seems to be inaccurate and impossible during first phases of software development cycle due to the insufficient information about the system at that period of time. This estimation is vital to companies and software developers since it can provide them with cost control delivery precision and other advantages [33]. Currently we have few model develop for SCE most of which function based on measuring LOC and FP. It is clear that size estimation accuracy directly affect cost estimation accuracy thus new alternatives such as Meta-heuristic algorithms can be a good option for SCE. The flowchart of the proposed model is illustrated in Figure (1).

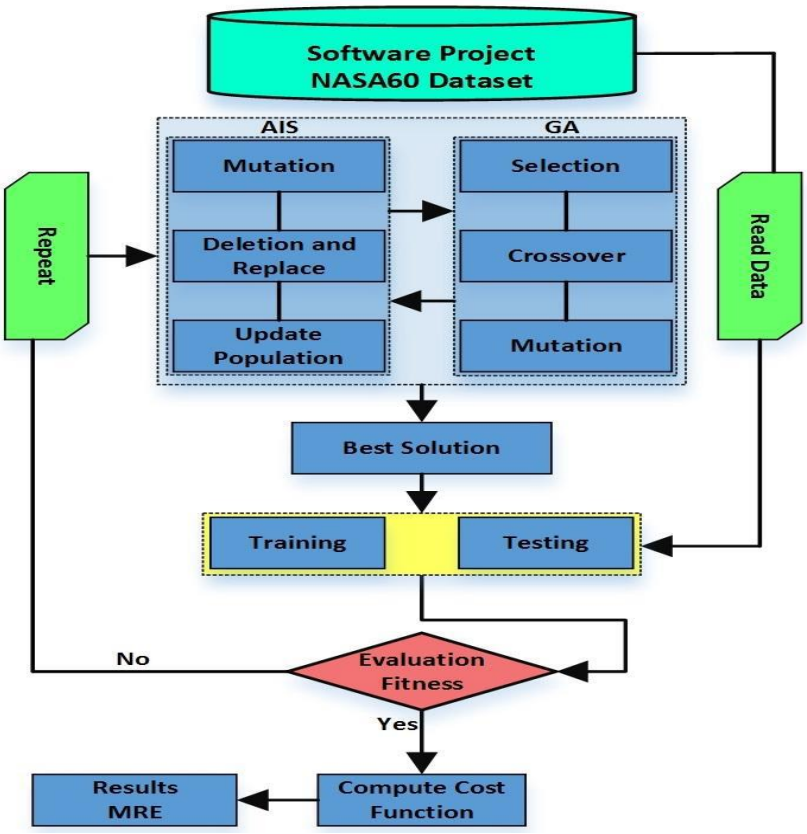



**Figure. 3. Flowchart of the Proposed Model**

A combination of AIS and GA operators have been used in the hybrid model to find the most optimized value for dataset aiming at estimating cost and effort. Fifteen EM factors were analyzed to obtain a better and integrated estimation. In the hybrid model the AIS algorithm tries to find the most optimized values based on LOC for the project and GA works on value training through re-optimization and selecting the most optimized values. At the end the objective function is evaluated and the values are tested on the dataset.

**Genetic Algorithms Overview:**

GAs simulates the survival of the fittest among individuals over consecutive generation for solving a problem. Each generation consists of a population of character strings that are analogous to the chromosome that we see in our DNA.

Each individual represents a point in a search space and a possible solution. The individuals in the population are then made to go through a process of evolution.

GAs is based on an analogy with the genetic structure and behaviour of chromosomes within a population of individuals using the following foundations:

- Individuals in a population compete for resources and mates.
- Those individuals most successful in each 'competition' will produce more offspring than those individuals that perform poorly.
- Genes from `good' individuals propagate throughout the population so that two good parents will sometimes produce offspring that are better than either parent.
- Thus each successive generation will become more suited to their environment.

**Search Space:**

A population of individualsare is maintained within **search space** for a GA, each representing a possible solution to a given problem. Each individual is coded as a finite length vector of components, or variables, in terms of some alphabet, usually the **binary** alphabet **{0,1}**. To continue the genetic analogy these individuals are likened to chromosomes and the variables are analogous to genes. Thus a chromosome (solution) is composed of several genes (variables). A **fitness score** is assigned to each solution representing the abilities of an individual to `compete'. The individual with the optimal (or generally near optimal) fitness score is sought. The GA aims to use selective `breeding' of the solutions to produce `offspring' better than the parents by combining information from the chromosomes.



The GA maintains a population of n chromosomes (solutions) with associated fitness values. Parents are selected to mate, on the basis of their fitness, producing offspring via a reproductive plan. Consequently highly fit solutions are given more opportunities to reproduce, so that offspring inherit characteristics from each parent. As parents mate and produce offspring, room must be made for the new arrivals since the population is kept at a static size. Individuals in the population die and are replaced by the new solutions, eventually creating a new generation once all mating opportunities in the old population have been exhausted. In this way it is hoped that over successive generations better solutions will thrive while the least fit solutions die out.

New generations of solutions are produced containing, on average, more good genes than a typical solution in a previous generation. Each successive generation will contain more good `partial solutions' than previous generations. Eventually, once the population has converged and is not producing offspring noticeably different from those in previous generations, the algorithm itself is said to have converged to a set of solutions to the problem at hand.

**Implementation Details:**

**Based on Natural Selection:**

After an initial population is randomly generated, the algorithm evolves the through three operators:

1. **selection** which equates to survival of the fittest;
2. **crossover** which represents mating between individuals;
3. **Mutation** which introduces random modifications.

**1. Selection Operator:**

- Key idea: give preference to better individuals, allowing them to pass on their genes to the next generation.
- The goodness of each individual depends on its fitness.
- Fitness may be determined by an objective function or by a subjective judgment.

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (1052-1063), Month: April - June 2015, Available at: www.researchpublish.com

**2. Crossover Operator:**

- Prime distinguished factor of GA from other optimization techniques
- Two individuals are chosen from the population using the selection operator
- A crossover site along the bit strings is randomly chosen
- The values of the two strings are exchanged up to this point
- If S1=000000 and s2=111111 and the crossover point is 2 then S1'=110000 and s2'=001111
- The two new offspring created from this mating are put into the next generation of the population
- By recombining portions of good individuals, this process is likely to create even better individuals


**3. Mutation Operator:**

- With some low probability, a portion of the new individuals will have some of their bits flipped.
- Its purpose is to maintain diversity within the population and inhibit premature convergence.
- Mutation alone induces a random walk through the search space
- Mutation and selection (without crossover) create a parallel, noise-tolerant, hill-climbing algorithms


**Effects of Genetic Operators:**

- Using selection alone will tend to fill the population with copies of the best individual from the population
- Using selection and crossover operators will tend to cause the algorithms to converge on a good but sub-optimal solution
- Using mutation alone induces a random walk through the search space.
- Using selection and mutation creates a parrallel, noise-tolerant, hill climbing algorithm

**The Algorithms:**

1. randomly initialize population(t)
2. determine fitness of population(t)
3. repeat

1. select parents from population(t)
2. perform crossover on parents creating population(t+1)
3. perform mutation of population(t+1)
4. determine fitness of population(t+1)

4. until best individual is good enough

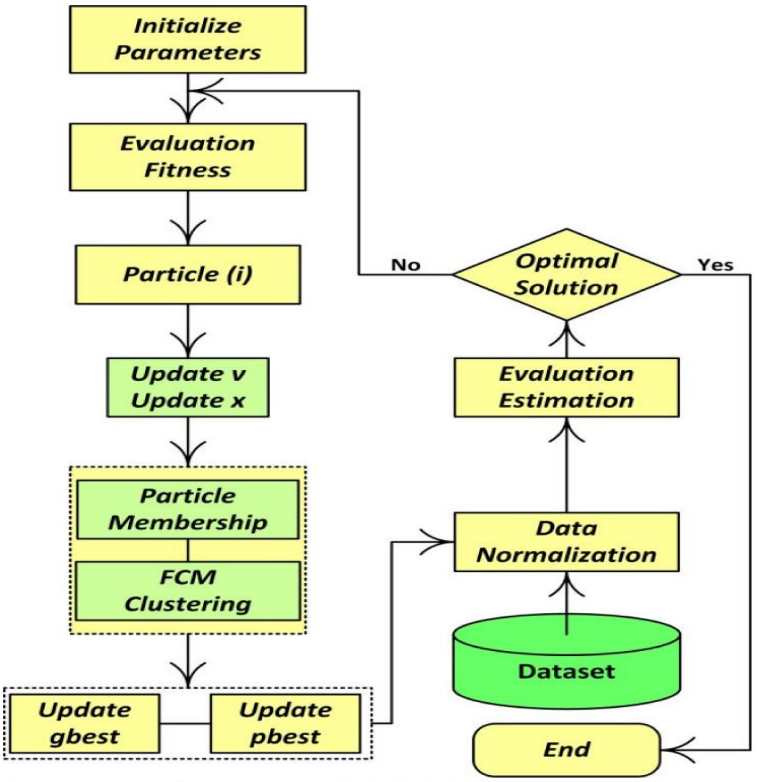
**Particle Swarm Optimization:**

PSO algorithm was introduced in 1995 inspired by the social behavior of birds that live in small and large groups20. PSO is a simulation of group social behavior of birds that search for food in an environment. None of the birds has information

ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (1052-1063), Month: April - June 2015, Available at: www.researchpublish.com

about the position of food but in each stage they know how far they are from the food. Based on this, the best way to find food is following the bird nearest to the food. The PSO algorithm is a population algorithm in which a few particles that are the solutions of a function or problem form a population. A population of particles moves in the problem space and based on their personal experience try to find the optimal solution in the search space. PSO algorithm as an optimization algorithm forms a population based search in which each particle changes its position in time. In PSO algorithm, the particles in move in a multi-dimensional search space of possible solutions. In this space an evaluation criterion is defined and the quality assessment of the problem solutions is achieved according to that. The variation of status of each particle in a group is influenced by its own experience and its neighbors' knowledge and the search behavior of one particle affects the other particles in the group. This simple behavior results in formation of optimal areas in search space. Therefore in PSO algorithm each particle informs other particles in a suitable way once it finds the optimal position and based on the achieved values for cost function, each particle decides according to a certain possibility, to follow other particles and the search in problem space is done based on the previous knowledge of the particles. This results in the particles not getting too close to one another and helps them efficiently solve the continuous optimization problems. In PSO algorithm first the individuals in the groups are accidentally created in the problem space and the search for the optimal answer begins. In the overall structure of the search each person follows the individual with the best fitness function, while not forgetting his own experience and follows the situation in which he has the best fitness function itself. Therefore each person in each algorithm iteration changes his future position according to two values, one being the best personal position so far (pbest) and the other the best position of all population so far and in fact the best pbest in the whole population (gbest). Conceptually, pbest for each person is in fact the biological memory of that person. Gbest is the general knowledge of the population and when people change their position according to gbest in fact they try to upgrade his own knowledge to the level of population knowledge. Conceptually the best particle group connects every particle in the group to one another.

FCM clustering algorithm is the most widely used algorithm in recognizing interrelated data in different clusters22. In FCM algorithm, at first random points are selected equal in number to the required clusters and then the data are assigned to one of these clusters according to the proximity and the clusters are formed. By repeating this procedure new centers can be calculated for the data through calculating the mean in each iteration and then the data are reassigned to new clusters. This procedure is repeated until no variation in the data is observed. In FCM clustering algorithm there is an objective function which defines the distance between the data.



**1. The swarm population is randomly generated**
**2. Evaluation Fitness**
**3. Repeat**
**3.1. Update position of each particle**
**●Local best**
**●Global best**
**3.2. Change the velocity and position**
**3.3. Cluster the particles using FCM**
**Calculate the cluster centers for each particle**
**Choose the optimum number of cluster particles**
**Update cluster**
**Update gbest and pbest**
**4. Data Normalization**
**5. Evaluation Estimation**
**6. Testing Dataset**
**7. until (Maximum Iteration Number)**
**8. Display Results**

## IV. CONCLUSIONS AND FUTUTRE WORK

**CONCLUSION:**

The SCE is one of the most sensitive, complicated and inevitable issues in the process of developing software. In the last three decades a significant growth was observed in using different types of SCE. In this increasing process it is possible that in fact in all cost estimation models, the major objective of software development cost reduction and easy and fast scheduling. The inherent ambiguity of the software development project results in the fact that there is a not high expectation for precise estimation. Algorithmic models in the field of estimation and simulation need numerous input parameters whose precise determination is difficult and their measurement also needs a lot of time and cost. In this article the hybrid models for SCE using PSO algorithm are presented. The test results have showed that the PSO-FCM and PSO-LA hybrid models have higher performance compared to COCOMO model and achieve higher accuracies in SCE. Because of approximate values and error estimations in hybrid models it should be kept in mind that in all using artificial intelligence can reduce error rate. According to the results of this paper it could be said that hybrid algorithms based on PSO can significantly improve the accuracy of the management decision making in the development of software projects.

**FUTURE WORK:**

This study hopes that more accurate estimation can be achieved combining COCOMO model with other meta-heuristic algorithms in future.

## REFERENCES

[1] Paternoster, N. Giardino, C. Unterkalmsteiner, M. Gorschek, T. Abrahamsson P. Software development in startup companies: A systematic mapping study, Information and Software Technology, 56(10): 1200-1218, 2014.

[2] Maleki, I. Ebrahimi, L. Gharehchopogh, F.S. A Hybrid Approach of Firefly and Genetic Algorithms in Software Cost Estimation, MAGNT Research Report, 2(6): 372-388, 2014.

[3] Maleki, I. Gharehchopogh, F.S. Ayat, Z., Ebrahimi, L. A Novel Hybrid Model of Scatter Search and Genetic Algorithms for Software Cost Estimation, MAGNT Research Report, 2(6): 359-371, 2014.

[4] del Bianco, V. Lavazza, L. Liu, G. Morasca, S. Abualkishik, A.Z. Model-based early and rapid estimation of COSMIC functional size-An experimental evaluation, Information and Software Technology, 56(10): 1253-1267, 2014.

[5] KHALIFELU, Z.A. GHAREHCHOPOGH, F.S. "A New Approach in Software Cost Estimation Using Regression Based Classifier", AWERProcedia Information Technology & Computer Science Journal, Vol. 2, pp. 252-256, December 2012.

[6] Pendharkar, P.C. Probabilistic Estimation of Software Size and Effort, Expert Systems with Applications, 37(6): 4435-4440, 2010.

[7] Mazhelis, O. Tyrväinen, P. Frank, L. Vertical Software Industry Evolution: The Impact of Software Costs and Limited Customer Base, Information and Software Technology, 55(4): 690-698, 2013.

[8] Li, Y.F. Xie, M. Goh, T.N. A Study of Project Selection and Feature Weighting for Analogy based Software Cost Estimation", Journal of Systems and Software, 82(2): 241-252, 2009.

[9] KHALIFELU, Z.A. GHAREHCHOPOGH, F.S. A Survey of Data Mining Techniques in Software Cost Estimation, AWERProcedia Information Technology & Computer Science Journal, Vol. 1, pp. 331-342, 2012.

[10] Boehm, B.W. Software Engineering Economics, Prentice-Hall, Englewood Cliffs, New Jersy, 1981.

[11] Boehm, B.W. Software Cost Estimation with COCOMO II, Prentice Hall PTR, Englewood Cliffs, New Jersy, 2000.

[12] Conte, S.D. Dunsmore, H.E. Shen, V.Y. Software Engineering Metrics and Models, The Benjamin/Cummings Publishing Company, Inc, Menlo Park, CA, p. 214, 1986.

[13] Albrecht, A.J. Gaffney, J. Software Function, Source Lines of Code, and Development Effort Prediction: a software science validation, IEEE Transactions on Software Engineering SE, 9(6), pp. 639-648, 1983.

[14] Gharehchopogh, F.S. Talebi, A. Maleki, I. Analysis of Use Case Points Models for Software Cost Estimation, International Journal of Academic Research, Part A, 6(3), 118-124, 2014.

[15] Maleki, I. Ebrahimi, L. Jodati, S. Ramesh, I. Analysis of Software Cost Estimation Using Fuzzy Logic, International Journal in Foundations of Computer Science & Technology (IJFCST), 4(3), 27-41, 2014.

[16] Farmer, J. Packard, N. Perelson, A. The Immune System, Adaptation and Machine Learning, Physica D, Vol. 2, pp. 187-204, 1986.

[17] De Castro, L. Timmis, J. Artificial Immune Systems: A New Computational Approach, Springer-Verlag, London, UK, September 2002.

[18] Holland, J. Adaptation in Natural and Artificial Systems, University of Michigan, Michigan, USA, 1975.

[19] Gharehchopogh, F.S. Maleki, I. Farahmandian, M. New Approach for Solving Dynamic Traveling Salesman Problem with Hybrid Genetic Algorithms and Ant Colony Optimization, International Journal of Computer Applications (IJCA), 53(1), 39-44, 2012.

[20] Sivanageswara, G. Rao, Phani Krishna, Ch.V. Rajasekhara Rao, K. Multi Objective Particle Swarm Optimization for Software Cost Estimation, Advances in Intelligent Systems and Computing, Vol. 248, pp. 125-132, 2014.