International Journal of Computer Applications (0975 8887)
Volume 96 - No. 21, June 2014

Software Project Scheduling by AGA

Dinesh Bhagwan Hanchate
Comp. Engg. Deptt.
V.P.'s College Of Engg.,
Baramati, Pune

Rajankumar S. Bichkar
Prof. (E & Tc) and Dean
G. H. R. C. O. E. M., Wagholi,
Pune, India.

ABSTRACT

This paper proposes general techniques for adapting operators in SGA for software project scheduling problem. The use of adaptive of crossover and mutation gives chance to control the diversity. Adaptive nature also tends to give convergence in the complex solution. Crossover and Mutation probability changes accordingly the change in the fitness values. High fitter is kept in the next pool. AGA(Adaptive genetic algorithm) converges to sub-optimal solution in fewer generation than SGA. In this paper, we consider skilled employees as an important resource to calculate the cost of the project along with some constrains of tasks. The paper gives a near-optimal estimated cost of project by using AGA. Our algorithm employs adaptive approaches for calculation of fitness of individuals, crossover rate and mutation rate. The paper also considers the aspects of head count, effort and duration calculated by COCOMO-II.1999. These parameters are used to verify the fitness of each chromosome to get estimated cost by AGA closer to the cost estimated by COCOMO-II.

General Terms:

Software Project Management, Machine learning

Keywords:

AGA, COCOMO-II, Software Cost Estimation, Project Scheduling.

1. INTRODUCTION

1.1 Natural evolution and GA

Natural evolution is discussed and expressed by one of its first proponents, Charles Darwin. His theory of evolution was based on four primary axioms [12].
⋆ An offspring has many of the characteristics of its parents. This axiom implies that the population is stable.
⋆ There are variations in characteristics between individuals that can be passed from one generation to the next.
⋆ The third axiom is the only a small percentage of the offspring produced survive to adulthood.
⋆ Survival of offspring depends on their inherited characteristics.
These all axioms and presumptions together imparts the theory of natural selection.
Another set of biologically-inspired methods are Genetic Algorithms (GAs). They derive their inspiration from combining the concept of genetic recombination with the theory of evolution and survival of the fittest members of a population [7]. The learning process devises better and better approximations to the optimal parameters, starting from a random set of candidate parameters. The GA is primarily a search and optimization technique. The genetic algorithm is a one of the family of evolutionary algorithms. Darwin discovered that species evolution based on two components: the selection and reproduction. The selection provides a reproduction of the strongest and more robust individuals, while the reproduction is a phase in which the evolution run.
The behavior of the GA depends on how we get the values of p_c and p_m. There are a various ways being told in regarding choosing p_c and p_m, by K. A. DeJong [25] [18].These are inadequate as the choice of the optimal p_c , and p_m becomes specific to the software problem under consideration. Grefenstette has formulated the problem of selecting p_c and p_m, as an optimization problem in itself.A theoretical comparison of randomized and genetic optimization algorithms concluded that many GAs are characterized by higher probability of finding good solutions than randomized algorithms, as long as the solution space fulfills several restrictions. These restrictions however are weak and hold for almost any choice of the genetic operators [13].
It is important to prevent promising individuals from being eliminated from the population during the application of genetic operators. To ensure that, the best chromosome is preserved, elitist methods copy the best individual found so far into the new population [25] [46]. However, elitist strategies tend to make the search more exploitative rather than explorative and may not work for problems in which one is required to find multiple optimal solutions [46]. In elitist strategy, the offspring have to compete with the parents to gain admission for next generation of GA [52]. The outstanding advantage of this environment is it always preserve the best solutions in every generation. Discussions on exploitation and exploration trade-off (by F. van den Bergh) has initiated the idea to investigate tournament and roulette wheel schemes other than deterministic in elitism strategy.

1.2 Related work

The approaches to project scheduling can be summarized as follows:
1.Search for optimal solutions using integer programming, dynamic and binary programming, branch and bound techniques, and
2.Search for suboptimal solutions using heuristic algorithms, including: Specialized heuristics; and Artificial intelligence methods are also exploited in the form of expert systems, ANN (Artificial

neural networks), and hybrid systems.
The scheduling problem is usually described as the task of integer programming. In such tasks, the vector of decision variables usually takes the form of a binary vector Brucker et al. 1999; Kasprowicz 2002; Kolish and Padman 1997; Marcinkowski 1990; Weglarz 1981. Precise procedures of single-criterion optimization of schedules are mainly based on the branch and bound method Dorndorf et al. 2000; Kasprowicz 2002; Weglarz 1981. To solve serious practical problems using precise algorithms is impossible because of the length of time needed for the calculations and the limited memory capacity of computers Marcinkowski 1990; Slowinski et al. 1994; Weglarz 1981.
Thus several approximation methods employing the heuristic approach have been conceived. The methods can be divided into two groups: specialized heuristics and meta-heuristics. Specialized heuristics can be used to solve only one optimization problem. Priority heuristics is among the most well known heuristics solving scheduling problems which is usually available in the project scheduling software. Priority heuristics are of two phases. The first phase prepares and arranges priority lists of processes according to decreasing values of priorities. The second phase calculate the start and finish times of these processes so as to keep to all the constraints in. In this phase, one of the two methods of tasks scheduling is used: parallel or serial, which differ in how they solve resources conflicts. The second phase is considered in our approach to have parallel tasks which are independent with each others. The works of Shanmuganayagam in 1989, Tsai and Chiu in 1996, and Ulusoy and zdamar in 1995 uses priority heuristics for scheduling projects and resources allocation. We used priority scheduling with respect to TPG only. Priority schedule may reduce the results that' why Khamooshi in 1996 modified the existing approach to establish process priorities. The procedure Khamooshi worked out and in dividing a project into parts and using a different priority rule for each part. He presents this approach in the form of a dynamic programming model. Slowinski et al. 1994 suggested employing a cluster of many rules instead of one priority rule, and then choosing the best one. We clustered the rules in terms of Hard and Soft constraints in our approach. To solve single-criterion optimization problems in scheduling projects, metaheuristic algorithms can also be used. They define only a certain pattern of optimization procedure, which must be adapted for particular applications software project scheduling by ACO (Jing Xiao, Xian-TingAo, YongTang 2012).
The most frequently used metaheuristic methods are taboo search, simulated annealing, and evolutionary algorithms. Actually, Neighborhood local search algorithms include simulated annealing and taboo search method (Sampson and Weiss 1993). They searched the feasible area solutions going from a current solution to a neighboring one. The natural imitating processes used in local search methods are also used in evolutionary algorithms.

1.3 Concerned readings and inspiration

The work done in fields, domains and sub-domains produced in various related papers are significantly carried out by some mean of related areas like genetics, GA, adaptiveness, software engineering, data structure required for GA, project management and interdisciplinary work done by researchers and authors. The readings of all the following papers is done for doing the study of various angles in the proposed approach. The work done by Mark [31], Dark [10], Sahani [22], Uyar [44], Forest [35], Imtiaz [24], Hayenes [20],Fogarty [16], Back [2], Yang [45], Therens [51] [50], Macheal and Shurva [**?**], Alba [14], Pinedo [41], Tom [37], Keightley [27], Charles [3], Parag [39], Thorat and Ambole [1], Zhang [57], Futuyam [17] and Jurgen [21].

1.4 Evolutionary algorithms

Evolutionary algorithms are classified into include evolution strategies, classifier systems,genetic algorithms, evolutionary programming, genetic programming.The results of research in this field are usually not classified according to an individual method but are generally described as evolutionary algorithms (Michalewicz 1996). Evolutionary algorithms work as computer systems for solving problems according to the rules observed in the evolution of live organisms. The rules involve system structure and the organisms ways function and adapt to existing conditions. A feature of this approach to solving optimization problems is creating a population of individuals representing solutions in a form of a chromosome. As in nature, better-adapted individuals more effective solutions which stand a better chance of survival and development.
The evolutionary algorithms are used to solve optimization problems in many branches of industry. A number of examples of their application, such as
♯ software project scheduling on timeliness GA (Carl chang 2010).
♯ basis the optimization of structures (Koumousis and Georgiou 1994), engineering and design (Grierson and Pack 1993),
♯ selection of equipment for earth-moving operations (Haidar et al. 1999).
Some studies show that evolutionary algorithms have a considerable potential to solve many project scheduling problems efficiently. ♯ For e.g., Li (1997;1999) used genetic algorithms to facilitate the time-cost optimization, and Hegazy 1999 applies them to the optimization of resource allocation and leveling.
♯ Leu and Yang 1999 developed a multi-criteria optimization model for construction scheduling based on a genetic algorithm, which integrates the time-cost trade-off model, the resource-limited model, and the resource-leveling model.
Some authors such as Padman 1997; Michalewicz 1996 classified evolution algorithms based on AI. Some experts like Adeli and Karim 1997, Kanet and Adelsberger 1987 solved the scheduling problem from ANN, expert systems apart from evolutionary algorithms.
Carl Chang proposed a tracking mechanism in the SPMNet [6]. He kept track of all the events through SDLC. An automatic technique based on genetic algorithms was introduced to determine the optimal resource allocation in 1994 by him. He and their co-authors calculated the total time and cost of a project dependant on the information generated from GA. SM (Software manager) may be able to predict the future states of a project. Runwei CHENG and Mitsuo GEN (1994) suggested approach wich can significantly improve the performance of evolution program both in term of the speed and the accuracy. Marek Pawlak (1995) presented an evolution program for project planning to implement an optimisation resource demand. The GA simulation based approach was demonstrated by Julia Pet (1995) with stochastic task durations using a multiple RCPS (resource constrained project scheduling) problem
.

Because practical application of precise methods is limited by the complexity of practical problems and imperfection of heuristic methods, the writers search for optimal and suboptimal project schedules using evolutionary algorithms. This approach proved to be appropriate for solving scheduling problems and relatively simple in computation. Even though, the method proposed by the some authors do not provide the optimal solution, the results are close to the optimum and can be obtained in a short time. Because, evo-

lutionary algorithms may be easily adapted to solving any type of problems, the proposed AGA method is versatile and allows defining the case conditions and constraints freely [29].
In order to adjust a genetic algorithm [34] to the optimization problem it tackles, some kind of parameter control is usually employed [33]. Adaptive control [26] uses feedback from the search to determine how the parameter values are to be altered. Self-adaptive control [?], puts additional information into the representation in order to provide some guidance to the operators. To date, most efforts on parameter control focused on adapting only one aspect of the algorithm at a time. The most comprehensive combination of forms of control [33] was offered in, where the adaptation of mutation probability, crossover rate and population size were combined. Other forms of adaptive systems for genetic algorithms can be found in [28] [36] . We proposed here a adaptive of a genetic algorithm which is able to dynamically adapt both the rate and the behavior for each of its operators.
All Genetic algorithms are a classified into stochastic type of optimization methods inspired by the principles of natural evolution. The promising research area in this area is adaptation of strategy parameters and genetic operators. For getting optimum solutions, different adaptive techniques has been used to guide search of GAs. A key component of GAs is mutation which is a variable GA operator to create diversity in GAs. There are several adaptive mutation operators in GAs, including population level adaptive mutation operators and gene level adaptive mutation operators. The experimental results of [24] show that the gene level adaptive mutation operators are usually efficient than the population level adaptive mutation operation.
The operator adaptation techniques in GAs can be classified into three categories, i.e., population level, individual level, and component level adaptation [40]. Operator adaptation depends on how operators are updated. At the population level, parameters are adapted globally by using the feedback information from the current population. Individual level adaptation changes parameters for each individual in the population. Component level adaptation is done separately on some components or genes of an individual in the population [40].
Adaptation of strategy parameters and genetic operators has become an important and promising area of research on GAs. Many researchers are focusing on solving optimization problems by using adaptive techniques, e.g., probability matching, adaptive pursuit method, numerical optimization, and graph coloring algorithms [49, 52, 38]. The value of parameters and genetic operators are adjusted in GAs. Parameter setting and adaptation in mutation was first introduced in evolutionary strategies [47]. The classification of parameter settings has been introduced differently by the researchers [11, 48]. Basically, there are two main type of parameter settings: parameter tuning and parameter control. Parameter tuning means to set the suitable parameters before the run of algorithms and the parameters remain constant during the execution of algorithms. *Parameter control means to assign initial values to parameters and then these values adaptively change during the execution of algorithms.* According to [11], parameters are adapted according to one of three methods: deterministic adaptation adjusts the values of parameters according to some deterministic rule without using any feedback information from the search space; adaptive adaptation modifies the parameters using the feedback information from the search space; and self-adaptive adaptation adapts the parameters by the GA itself.
There are two main groups of adaptive mutation operators, one group are the population-level adaptive mutation (PLAM) operators and the other are the gene-level adaptive mutation (GLAM) operators. Many researchers have suggested different static mutation probabilities for GAs. These static mutation probabilities are derived from experience or by trial-and-error.

Table 1. Athours and suggested P_m.

Author	Suggested P_m
De Jong	0.001
Schaffer	[0.001, 0.005]
Back	1.75/(N $\times$ L1/2)

where, N means the population size and L denotes the length of individuals. This equation is based on Schaffers results [?]. In [11], it is suggested that P_m = 1/L should be generally optimal. It is very difficult, though not impossible, to find an appropriate parameter setting for P_m for the optimal performance.
In the simple GA, the penalty function that is used to convert the constraints problems to unconstraint ones, and genetic operators, such as cross-over, mutation, and elitism, so on that are used to explore the important regions of the search space are adopted. Since there is not a unique way to define the penalty scheme and genetic operators, different forms of these are proposed (Rajeev and Krishnamoorthy, 1994; Rajan, 1995; Krishnamoorthy et al., 2002). In contrast to classical penalty scheme and genetic operators having the values of the various coefficients treated as pre-defined constants during the calculation of penalty function, some enhancements in the GA have been made and proposed by the researchers (Nanakorn and Meesomklin, 2000; Chen and Rajan, 2000; Srinivas and Patnaik, 2000; Togan and Daloglu, 2006) in order to increase the efficiency, reliability and accuracy of the methodology for code-based design of structures.
In this paper, adaptive approaches are proposed for both the penalty function and crossover and mutation probabilities in order to relieve the user from determining any values that are needed prior the optimization and enhance the performance of the GA in optimizing SPSP. In the simple GA, the penalty function that is used to convert the constraints problems to unconstraint ones, and genetic operators, such as cross-over, mutation, and elitism, so on that are used to explore the important regions of the search space are adopted. Since there is not a unique way to define the penalty scheme and genetic operators, different forms of penalty scheme and genetic operators are proposed (Rajeev and Krishnamoorthy, 1994; Rajan, 1995; Krishnamoorthy et al., 2002). In contrast to classical penalty scheme and genetic operators having the values of the various coefficients treated as pre-defined constants during the calculation of penalty function, some enhancements in the GA have been made and proposed by the researchers (Nanakorn and Meesomklin, 2000; Chen and Rajan, 2000; Srinivas and Patnaik, 2000; Togan and Daloglu, 2006) in order to increase the efficiency, reliability and accuracy of the methodology used.In 1975, Holand ,De Jong, and in 1987 Ackley proposed one-point, N-point crossover and uniform crossover respectively.
The paper's outline is organized as follows.
◦ Section I, we discussed the GA & optimization, and the various techniques proposed in the literature and related work to overcome the problems.
◦ Section II describes GA and itś significance in optimisation.
◦ In section III,We light on the role of project manager, importance of time management in SPM.
◦ In section IV, we gave the utilisation of SEE (Software Engineering Economics) in our approach.
◦ In section V we formulated the SPSP with definition.

○ In Section VI, our approach of using adaptively varying probabilities of crossover and mutation for SPSP (software project scheduling problem) is focused.
○ In section VII, We have Operations, Mathematical Modeling And Adaptive Fitness for our approach.
○ We illustrated input and output to our problem in section VIII.
○ A discussion of results obtained, some conclusions of the study and future work still to be done are explained in section IX,X,XI respectively.

2. GENETIC ALGORITHM

All GAs are inspired by the biological evolution. Each individual represents binary strip called chromosome. Each element in the strip is called as gene where each individual shows the possible solution of CO (Constraints optimization) problem in our hand after evolution. GAs are powerful methods of optimization used successfully in different problems. Their performance is depending on the encoding scheme and the choice of genetic operators especially, the selection, crossover and mutation operators. A variety of these latest operators have been suggested in the previous researches. In particular, several crossover operators have been developed and adapted to the permutation presentations that can be used in a large variety of combinatorial optimization problems.
Genetic algorithms (GAs) are powerful search methods. GAs were first introduced by John Holland in 1960s in USA. Nowadays,GAs have been successfully applied for solving many optimization problems due to the properties of easy-to-use and robustness for finding good solutions to difficult problems [6]. The efficiency of GAs depends on many parameters, such as the initial population, the representation of individuals, the selection strategy, and the recombination (crossover and mutation) operators. Mutation is used to maintain the diversity of the entire population by changing individuals bit by bit with a small probability p_m. Usually, the mutation probability has a significant effect on the performance of GAs.

2.1 Simple Genetic Algorithm

```
Algorithm 1: Simple Genetic Algorithm
  1.Initialize population
  2.Evaluate population
  while convergence not achieved do
      a.Scale population fitness
      b.Select solutions for next population
      c.Perform crossover and mutation
      d.Evaluate population
  end
```

Fig. 1. Simple Genetic algorithm

GA is optimization search techniques useful in a number of practical problems. The robustness of Genetic Algorithm is method with it's behaviour to find the global optimum in a hypothesis. GA's random, directed is search for locating the globally optimal solution. Randomized and directed GA is specially for finding the globally optimal solution. GA is useful as tool for a genetic representation for the feasible solution. We get a population of encoded solutions. We have a fitness function that evaluates the optimality of each solutions. Genetic operators generate a new population from existing populations. The GA is iterative process of population evolution for sequenced generations. After and in between generation,each solution has fitness which gives to decide the next chromosomes to do the mating.Fitter is selected for meting and other are discarded from the pool. The fitter is selected by and from fitness values for next generation. The selected chromosomes then has to go for crossover and mutation operation. The main and important operation is crossover which makes a structured and randomized exchange of alleles with crossover possibility. Crossover can be done by P_c, crossover rate. Mutation does the flipping of the allele of a chromosome by mutation probability. The mutation plays important role to do this by restoring the lost of genetic material. Scaling is another operation which is useful for maintaining the steady selecting pressure in objective function.
It is an objective that GA should not converge towards the optimal solution by taking much of iterations. In this section, we discuss the role of the parameters in controlling the behavior of the GA. We also discussed the techniques proposed in the literature for optimization of SPS problem for enhancing the performance of GA . The significance of p_c and p_m in controlling GA performance has long been acknowledged in GA research [7]. The crossover probability p_c controls the rate at which solutions are subjected to crossover. The higher the value of p_c, the faster are the new solutions introduced into the population. As p_c, increases, however, solutions can be disrupted faster than selection so, typical values of p_c, are in the range 0.5-1.0.
Mutation is only a secondary operator to restore genetic material. Nevertheless the choice of p_m plays important role in GA performance and has been emphasized in DeJongs work [25]. Large values of p_m , transform the GA into a solely random search algorithm, while some mutation is required to prevent the premature convergence of the GA to suboptimal solutions. Typically p_m , is chosen in the range 0.005-0.05. We must thanks to DeJong for his the Efforts made to improve GA performance in optimization. DeJong introduced the ideas of overlapping populations and crowding in his work also. In the case of overlapping populations, newly generated offspring replace similar solutions of the population, primarily to sustain the diversity of solutions in the population and to prevent premature convergence. However,the crowding factor (CF) is introduced in this technique which has to be tuned to certain optimal GA performance. In all the techniques described above, no emphasis is placed on the choice of p_c and p_m. p_c , and p_m is still left to the user to be determined statically prior to the execution of the GA. The idea of adaptive mutations and crossover is already employed to improve GA performance. Our approach for finding SPSP (software project schedule Problem) also uses not only Crossover and mutation adaptive probabilities but also adaptive fitness, but in a manner different from these previous approaches.
In the next section, we discussed the motivation for having adaptive probabilities of crossover and mutation, and describe the methods adopted to realize them. In this paper, We used adaptive probabilities of crossover and mutation to see the effect of keeping simple track of diversity and taking linear load of convergence. Another advantage of our approach is to provide a solution to the problem of choosing the optimal values of the p_c , and p_m. (We referred crossover and mutation probability as p_c and p_m, respectively).

3. SOFTWARE PROJECT MANAGEMENT

3.1 Objectives

Each software project is initiated by the need of an organization to deliver new software products and solutions to the market in order

to produce profit, knowledge or something else of the organizations interest. Each software project has its own main objective which is defined at the very beginning. It is the task of the software project management process to ensure that software project main objective is achieved. There are some common things that most of the projects share among themselves, e.g.:
· Specific start and threshold date.
· Time schedule,Direct cost, Indirect cost, budget and quality constraints.
· Particular attention to get the target result.
Software Project management is the application of logical and algorithmic knowledge, skills, software tools and different techniques to meet the userś requirements of the particular project. Software Project management, knowledge and practices are best described in terms of their some of s/w engineering processes eg. waterfall model, RAD, Spiral model. These processes can be placed into five process groups and nine knowledge areas. These groups are Requirement engineering, Design and analysis, coding, testing and integration (closing). These areas are project integration management, project scope management, project time management,project cost management, project communications management, project risk management and project procurement management, project quality management, project human resource management.
There are three very important threads that we can find in almost all the software projects: Cost (Money), Schedule (time) ,and performance (quality). Two of them we may not exceed - Cost and Duration (Time) and the third one quality must be at least as required by the customers. Above mentioned processes are actually a processes of managing a number of others, mutually dependant sub-processes each having its own objective, but contributing to the main project objective. We will see in the next section about project managerś role and time management as this aspects are more focused and related to our SPSP.

3.1.1 **software Project Manager**. A Software PM is a person who is responsible for the project. He/she may lead the software development towards a successful or, sometimes unsuccessful software product. A Software PM must have a set of competencies that make him/her appropriate person for such a duty. A project manager, besides leading the project, should have well communications with other related instances involved in and around the project other than software related technical problems. He has to play a role of interface that the best software PM must play. He must keep common and important relations intact with project sponsors and users. Software PM is directly responsible to the project sponsor as the sponsor is the one who orders a project to be executed. Project sponsor usually defines some specific boundaries (e.g. budget, time, quality) and monitors them closely. That's why AGA approach is not only useful for the Software PM but for the quality oriented project as our approach considers some of the aspects of SEE. Users are those who usually specify the requirements and negotiate technical details of realization that are of high importance to them. When we talk about software development, a common situation is that the client orders some extensions of already existing systems, or wants some new systems that are compatible to other existing systems. In any case, very usual things are change requests during development or after finishing the original project. That's why we defined some flexible software engineering oriented constraints that has to be met during the scheduling of project.

3.1.2 Software project Time Management (SPTM). SPTM is a subset of project management that includes the processes required to ensure timely completion of the project. It consists of problem definition, TPG (Tasks precedence graph), event and activity sequencing, schedule (DUR) estimating, schedule development according to target schedule, and schedule control. Since, time is one of the most important management factor and parameter that must be obeyed during software project development (and in SDLC), it will involve a lot of planning and reviewing on monthly, weekly or even daily basis. We come across the term Milestone in SPTM. Defined and target Milestones shows Umbrella activities with WBS (work breakdown structure) points in the SPM. Milestones are generally associated with important baselines and results. They are defined using time and appropriate deliverables. In another words milestones define end of certain project development phase.

4. SOFTWARE COST ESTIMATION (SCE)

There are main seven steps that is generally adapted in SCE as per the SEE [4].
Step 1. Establish objectives.
Step 2. Plan for required data and resources.
Step 3. Write down software requirements.
Step 4. Work out in detail work (WBS) as we can.
Step 5. Use different independent techniques and sources.
Step 6. Compare and Iterate budgeted estimates.
Step 7. Followup.
Out of these seven step, we work out only on roughly on step 1 and 2 and Step 5. We tried to define the problem, requirements in terms of hard and soft constraints and schedule the project with one technique i.e. by AGA.

4.1 Effort, COCOMO Models

There are various COnstructive COst MOdels developed by Barry W. Boehm and others. Estimating effort of software is product of productivity and size of team. The unit of effort is man-month (MM) or person-month (PM) which is calculated in terms of KDSI (thousands of delivered source instructions) by following equation [4].
MM= 2.4 (KDSI)$^{1.05}$.
Also, the development schedule (TDEV) in months is given as
TDEV= 2.5 (MM)$^{0.38}$
The above equation is a basic model applicable to the large majority of software projects [4]. According to COCOMO model a man-month (MM) is equal to 152 hours of working time.
Table no.2 shows Effort and Duration for three different types of project as per the COCOMO model. We assumed the calculated efforts (by COCOMO Model) as one of the inputs to our project. Further calculation, evaluation and comparison with our model is done by following equations.

$$TDEV = 3 \times (MM)^{0.328} \quad (1)$$

$$HC = 0.666 \times (MM)^{0.672} \quad (2)$$

Above equations are also used for calculating the constrain factor, the effort and head count.

5. PROBLEM DEFINITION AND FORMULATION [?]

A project schedule is an assignment of the tasks to the 4Ps at particular duration by considering all the constrains of the project to get the optimal solution with optimum cost and time. Each task requires a set of skills and effort [9].

Table 2. Notations ,symbols and meanings.

Notations	Meaning
MM	man-months
PM	Person-Months
KDSI	Thousands of delivered source instructions
TDEV	Development time
HC	Head Count
GSCH	Schedule derived by our approach using GA
CD T_i	COCOMO Calculated duration
COD T_i	COCOMO Optimistic duration = 0.86 $\times$ CD
AGOD$_{T_i}$	Optimistic duration obtained by our AGA approach
CPD$_{T_i}$	COCOMO pessimistic duration of task T$_i$ =1.6 $\times$ CD
EF$_{avg}$	Stretch out effort of the project
EF$_{pa}$	The Pessimistic effort at analysis phase
EF$_{pd}$	The pessimistic effort at design phase
EF$_{pi}$	The pessimistic effort at implementation phase
EF$_p$	The pessimistic effort of whole project.
TPHC	Total project head count.
EMF	Effort multiplier factors
P$_c$	Crossover rate
P$_m$	Mutation rate
sth	Schedule number in generation
χ(AGSHD$_s$)	Modified Fitness of sth Schedule
FC$_s$ $\equiv$ F(AGSHD$_s$)	Fitness of sth Schedule
P$_s$	Total Penalty of sth schedule
ND$_s$	Normalised duration of sth schedule
SE$_{max}$	Maximum salary of employee among all employees
TCHC$_s$	Total COCOMO head count of sth schedule
AGSHD$_s$	Schedule obtained by our genetic approach of sth schedule
CPSHD $_s$	COCOMO pessimistic of sth schedule
COSHD $_s$	COCOMO optimistic of sth schedule
DP$_s$	Degree of penalty of sth schedule
P$_{avg}$	Average penalty of generation
P$_{max}$	Maximum penalty of generation
P$_{min}$	Minimum penalty of generation
NHCP$_s$,	Normalised head count Penalty
NTP$_s$,	Normalised total time Penalty
NITP$_{Ti}$	Normalised task incompleteness penalty
TNITP$_s$	Total Normalised Incompleteness Task Penalty of sth schedule

Table 3. Formulae for Effort and Duration according to the types of project.

Organic	MM = 2.4 (KDSI)$^{1.05}$
	TDEV= 2.5 (MM)$^{0.38}$

Semidetached	MM= 3.3 (KDSI)$^{1.12}$
	TDEV=2.5 (MM) $^{0.35}$

Embedded	MM=3.6(KDSI)$^{1.20}$
	TDEV= 2.5(MM) $^{0.32}$

Table 4. Tables and their description.

Fig. No.	Figureś Description
1	Athours and suggested Pm
2	Notations ,symbols and meanings
3	Formulae for Effort and Duration according to the types of project
4	Tables and their description
5	Figures and their description
6	A Sample example of solution in terms of Schedule representation
7	Input:Configuration file [9] showing mathematical notations for T10E5S4 as an example
8	Input:Task Properties for T20E5S4
9	Input:Employee Properties for T20E5S4
10	Output:Employee working times for T20E5S4
11	COCOMO and AGA durations for T20E5S4 Where, CODT$_i$ $\leq$ AGADT$_i$ $\leq$ CPDT$_i$
12	Output:Tasks duration in Months with employee for T20E5S4
13	Output:Xover Vs Avg Computation time for T20E5S4
14	Comparative chart of our and Ting-Tang approach for G2 Group
15	Comparative chart of our and Ting-Tang approach for G3 Group.
16	Comparative chart of our and Ting-Tang approach for G4 group.

Table 5. Figures and their description **T20E5S4 means 20 Tasks, 5 Employees, 10 Skills example but employees possesses 4-5 skills.**

Tab. No.	About Figure
1	Task Precedence Graph for T20E5S4
2	Generation vs Fitness
3	Generations Vs Task Completed T20E5S4
4	Generation Vs Critical Duration for T20E5S4
5	Schedule for first year for Task Precedence Graph for T20E5S4
6	Schedule for second year for T20E5S4
7	Schedule for Third year for T20E5S4
8	Graph shows assignment of employee, duration for each employee to tasks, critical path duration, total duration for T20E5S4
9	First part of Network diagram for T20E5S4
10	Second part of Network diagram for T20E5S4

• Let T be a set of tasks, T={T$_i$, i=0,....,n-1} where n is the number of tasks,
• Let E be a set of employees, E={E$_i$, i=0,....,e-1} where e is number employees,
• Let S be a set of skills, S={S$_i$, i=0,...,m-1}, where m is the total number of skills.
• Let ES be a set of skill of employees,
ES={ES$_i$,i=0,....,m-1} where m is the number of skills and
• EF be the effort required for the tasks in T,
EF={EF$_{T_i}$, i=0,...,n-1} where EF$_{T_i}$ is the effort required for task T$_i$.
• The skills required by tasks are represented by an n $\times$ m sized

task skill matrix i.e TS,
where TS={TS_{ij}, i=0,....,n-1, j=0,...,m-1}
Each elements TS_{ij} of task skill matrix S is either 0 or 1, depending on whether task T_i requires skill S_j as

$$TS_{ij} = \begin{cases} 1 & if\ Task\ T_i\ requires\ skill\ S_j. \\ 0 & Otherwise. \end{cases}$$

The dependence [39] between the tasks is given by task dependency matrix (TD) of size of n×n. Its elements are given as,

$$TD_{ik} = \begin{cases} 1 & if\ Task\ T_i\ depends\ upon\ task\ T_k. \\ 0 & Otherwise. \end{cases}$$

Finally, AGSHD is a n×e sized task assignment matrix of duration (in months) assigned to each employee on various tasks. The duration may be in years, months, quarters or weeks. TD matrix is obtained from task precedence graph (TPG). The Task Precedence Graph shows the precedence relation between the tasks, is an acyclic Graph, G(T,EG) where The T represents the set of all task nodes included in the project and EG is the set of edges between dependent tasks [**?**]. A sample TPG for 10 tasks, 5 employees and 10 skills is shown in Fig 1. There are various types of tasks in a project

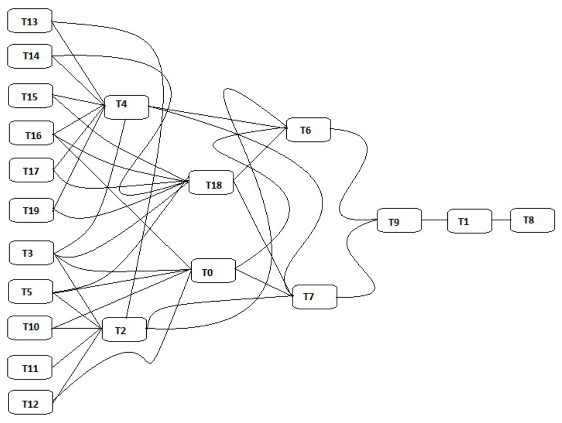


Fig. 2. Task Precedence Graph for T20E5S4

[55]. These are Start to Start (SS), Start to End (SE), End to Start (ES), and End to End (EE). In this paper, we have considered the tasks of SE type. Once a task starts for such tasks, it has to end without fail but by maintaining parallel or concurrent mechanism. Each task has associated with the optimistic as well as pessimistic values of effort and head count. Here, we have made the average of the pessimistic values of 3 SPM phases.

$$EF_{pa} = 2 \times EF \tag{3}$$

$$EF_{pd} = 1.5 \times EF \tag{4}$$

$$EF_{pi} = 1.25 \times EF \tag{5}$$

$$Ep_{avg} = 1.583333 * EF \tag{6}$$

Where, EF_{avg} is stretch out effort of the project (project maximum effort),

⋆ EF_{pa} is the Pessimistic effort at analysis phase,
⋆ EF_{pd} is the pessimistic effort at design phase,
⋆ EF_{pi} is the pessimistic effort at implementation phase and
⋆ EF_p is the pessimistic effort of whole project. *We calculated all the above corresponding duration CPD,COD using equation1* We have taken average effort as threshold value and hard constraint in our problem. We have 25% managerial margin to work for scheduling software project apart from CMM (Capability Maturity Model) model under consideration. CMM's one of the principles says that 30 PC FSP should be kept as extra staff for substitution in case of critical situation. This paper gives scheduling of software project by considering some hard and soft constraints which is described in next subsection.
We used an indirect constraints handling i.e. objectives optimization by satisfying the constraints. In general, penalties are given for violated constraints. Some GAs allocate penalties for wrongly instantiated variables also for the distance up to a feasible solution. The generality of penalty gives reduction of the problem to simple optimization. The described and defined constraints for our SPSP (Software Project Scheduling Problem) are given and listed bellow.

5.0.1 Hard constraints. The tasks in given SPS problem are to be assigned to employees subject to the following hard constraints [5].

(1) All skills must be matched. The employee must have skills required for the tasks to be done.
(2) Task precedence graph must be satisfied during an assignment of a task to an employee.

5.0.2 Soft constraints. Also the following soft constraint are considered to obtain a better schedule.

(1) At least 80 PC of task's head count should be as per values calculated by COCOMO-II.2000 and may differ at most ± 1 for each task head count. This is managerial adjustment, normally given by industry to the PM, to add or remove one employee to maintain the quality of the project. But, This adjustment is directly related to effort.
(2) Cost of the project should be below the average cost calculated by COCOMO-II.2000 calculation.
(3) Employee may not be overloaded more than 50% of his/her capacity.

$$OL_{Ei} \leq 0.5 \times C_{Ei} \tag{7}$$

OL and C are the overload and capacity respectively. Capacity of employee is how much employee can do quality work in one unit time. (The capacity is described in section IIIA).
(4) Work should be equally distributed amongst employees as far as possible.
(5) Equal importance should be given to project cost and duration.
(6) For maintaining the quality, each duration of task should be nearer to the optimistic duration (COD).
(7) The total number of employees for project tenure should be in between 80% and 100% of Head count calculated by COCOMO-II.

$$\mathit{120\%\ of\ TPHC} >= \sum_{i=1}^{n} HC_{Ti} \geq \mathit{80\%\ of\ THC}. \tag{8}$$

TPHC is total project head count.

(8) The task duration should not exceed the pessimistic duration (CPD$_{T_i}$).

$$TDEV_{T_i} = 3 \times EF_{T_i}^{0.328} \tag{9}$$

$$\sum_{j=1}^{e} SCH_{Ti\,,\,Ej} \leq CPD_{T_i}. \tag{10}$$

where, CPD is pessimistic duration calculated using COCOMO-II.2000.
CPD$_{T_i}$=1.6 $\times$ TDEV$_{T_i}$ (according to equation no.6 and analogous to it).

(9) Total duration computed by GA of all tasks should be above 86 % of duration calculated by COCOMO-II.2000 calculation.

$$\sum_{i=1}^{n}\sum_{j=1}^{e} AGSHD_{Ti\,,\,Ej} > 0.86 \times \sum_{i=1}^{n} COD_{T_i}. \tag{11}$$

Above constraint is taken by considering personnel effort multiplier factors (PEMF), Where PEMFs are the effort multipliers defined by COCOMO-II as these factors lies in between 0.86 to 1.56 as per the quality and skill proficiency of an employee. Our problem takes **Normal** scale which gives 1.0 as scale of (multiplication factor) effort multipliers as all employees are considered at nearly equal level. Cost of the project should be in between the cost calculated by normal scale of COCOMO-II.2000 and adjusted scale of COCOMO-II calculation. Quality drivers (also called as effort multipliers) of all employees for personnel properties assumed are not above the maximum adjusted scale of 1.6 i.e. adjustment factor of all the employees are considered in the range of 1.0 to 1.6, where adjustment factor is the change in the some of the effort multiplier factors (EMF) [23].

6. PROPOSED AGA APPROACH

6.1 Motivations

A software project scheduling is CSP(Constrained Satisfying problem) with objective optimisation. We require pushing energy to meet optimal solution after getting the optimal space. Not only this, but also, we need another property of finding new space of solution in optimal search. Interestingly, The trade-off between these properties is hiddenly packaged in values of p$_c$ and p$_m$ and also in crossover type. Usually, in practice,we take large values of p$_c$ (0.5-1.0) and small values of p$_m$(0.001-0.05), In our approach, we put this trade-off by varying p$_c$, and p$_m$ adaptively in response to the fitness values of chromosomes. p$_c$, and p$_m$ are increased when the population is about to get stuck at a local optimum and are decreased when the population in solution space is scattered.
To vary p$_c$, and p$_m$, adaptively, for preventing premature convergence of the GA to local optimum. We should able to identify whether the GA is on right optimum path of convergence.
For detecting the convergence ,we have to see average fitness value $\overline{f}$ of the population in relation to the maximum fitness value f$_{max}$ of the population. The minimum the value of f$_{max}$ - $\overline{f}$, more is the convergence to an optimum solution, it means a population scattered in the solution space.p$_c$ and p$_m$ has to be increased if GA converges to local optimum when f$_m ax$ - $\overline{f}$ decreases. P$_c$, and p$_m$ has to be increased if solutions come towards scattered pool.

$$p_m \quad \alpha \quad f_{max} - \overline{f} \tag{12}$$

$$p_c \quad \alpha \quad f_{max} - \overline{f} \tag{13}$$

We use the difference in the average and maximum fitness values, f$_{max}$ - $\overline{f}$, as a benchmark for detecting the convergence. † $\overline{f}$ **is also denoted by f$_{ava}$ in adaptive mutation, crossover in next section.** It has to be observed in the above expressions that p$_c$ and p$_m$ do not depend on the fitness value of any particular solution, and have the same values for all the solutions of the population. Consequently, solutions with high fitness values as well as solutions with low fitness values are subjected to the same levels of mutation and crossover. When a population converges to a globally optimal solution (or even a locally optimal solution), p$_c$, and p$_m$, increase and may cause the disruption of the near-optimal solutions. The population may never converge to the global optimum.
To overcome the above-stated problem, we need to preserve good solutions of the population. This can be achieved by having lower values of p$_c$ and p$_m$ for high fitness solutions and higher values of p$_c$ and p$_m$ for low fitness solutions. This can be seen and realised in the B.4 and B.6 subsections of VII. the high fitness solutions aid in the convergence of the GA, the low fitness solutions prevent the GA from getting stuck at a local optimum. The value of p$_m$, should depend not only on f$_{max}$ - f, but also on the fitness value f of the solution. Similarly, p$_c$ should depend on the fitness values of both the parent solutions. The closer f is to f$_{max}$, the smaller p$_m$, should be, i.e.,p$_m$, should vary directly as f$_{max}$ - f. Similarly, p$_c$, should vary directly as f$_{max}$ - f, where f is the larger of the fitness values of the solutions to be crossed.

```
Algorithm 2: Adaptive Genetic Algorithm
AGA( PS,$P_c$,$P_m$)
// Initialize $0^{th}$ Generation,Where,individual i=1:PS
i =1;
$G_i$ : $i^{th}$ individual randomly-Generated of Generation G.
//where G = 1:g, g: Max Generation
//Evaluate $G_i$
Get $\chi(G_i)\ \forall$ i $\varepsilon$ G
while convergence not achieved or $G \neq g$ do
    1. adaptive Elitist Crossover:
    Select PS $\times$ $P_c$ members of G
    Pair them up: Produce Offspring
    Select 2 Highest fitters and insert it into G+1
    along with other individuals.
    2. Adaptive Mutation
    Select PS $\times$ $P_m$ members of G
    Invert individual selected bit each.
    3. Get $\chi(G_i)\ \forall$ i $\varepsilon$ G
    4. G= G+1
end
```

Fig. 3. Adaptive Genetic algorithm

6.2 Steps in AGA

The successive steps of the AGA is described in general below.

6.2.1 Step 1. Initiation.. Initiation consists in creating an initial population specified number of individuals chromosomes. The writers use individual representation in the form of gene strings containing information about methods and values of processes priority. The initial population is created randomly. Particular genes assume values chosen randomly with equal probability from their variability interval. Activity priorities, allocated randomly in the initial population, are modified in consecutive steps of the algorithm until a solution that corresponds to the shortest duration of the project is obtained. Therefore, the AGA enables the user to find optimal values of priorities that determine the sequence of allocating tasks to employees, to activities. The algorithm is thus a tool that may help PM in their everyday work of making decisions and setting priorities.

6.2.2 Step 2. Individuals assessment.. This procedure is used to calculate project duration, and thus it enables chromosomes feasible solutions assessment. To assess the solutions generated by AGA, the authors worked out the heuristic algorithm for tasks to employee allocation and calculating the shortest project duration presented in the next section.

6.2.3 Step 3. Protection of the best individual.. The individual chromosome from the initial population for which the objective function value is the best the shortest project duration is remembered. The best individual protection also-called exclusive strategy is a special additional reproductive procedure. The best adapted individual, among all from former generations, does not always pass to a new population. Exclusive strategy is used as the protective step against the loss of that individual.

6.2.4 Step 4. Calculating value of individuals adaptive fitness function.. AGAs are used to look for the best adapted individuals for which the fitness function value is the highest. The study focuses on finding the solutions of minimization problems. In this case, it is necessary to convert the minimized objective function into maximized fitness function. The calibrating fitness function prevents premature convergence of the evolutionary algorithm, which would result in finding a local optimum and not a global one.

6.2.5 Step 5. Checking the termination condition.. The action of the algorithm can be stopped in two cases: after performing a specified number of iterations when the number of the current generation is greater than the maximum value assumed, and when, after some number of iterations, there are no better solutions than in previous generations. If the termination condition is not met, a selection of individuals is carried out as the next step.

6.2.6 Step 6. Selection procedure (Elitism).. Chromosomes selection consists in choosing individuals that will take part in producing offspring for the next generation. Chromosomes having the highest fitness function value are the most likely to produce new individuals. The last step is repeated for each individual in the population.

6.2.7 Step 7. Crossover (Adaptive) . The task of crossover is to recombine chromosomes by exchanging strings of genes between parents chromosomes. The different crossovers are employed in the study. Strings of genes in the parents chromosomes ahead of the point of crossing are not changed, only genes behind that point are exchanged between parents.

6.2.8 Step 8. Mutation (Adaptive). Mutation involves random change of one or more genes of the selected chromosome, with probability equal to mutation frequency. Calculation of the fitness function value for each individual in a new generation, the best individual protection, selection procedures, crossover, and mutation are repeated cyclically until the termination condition of the algorithm is met. Then the result of algorithms action, i.e., the solution to the problemthe way of using skilled employee, the project duration, and beginning and finishing of each task is given. The best solution corresponds to the individual having the lowest value of the assessment function the shortest or minimal project duration.

6.3 The Critical Path Procedure

Following is the CPP is used in our approach to find and compare them with previous individual's CPT to get optimum duration AGAD.

```
Algorithm 3: Critical Path Algorithm
// SN : Start Node,ULN:Unlebelled Node Set,
//LN:labelled Node Set,PN: Predecessor Node Set,
//EST:Earliest Starting Time,LFT:Latest Finishing
//Time,$D_N$: Duration of activity
//$PN_N$ is the set of predecessor nodes of N.
1. SN $\leftarrow$ (0,0),LN = $\phi$,ULN = All ULN nodes
2.while $ULN \neq \phi \bigwedge \forall\ ULN_N\ (PN_N \subseteq LN)$ do
    a.$EST_N$= max $LFT_i$ Where i$\epsilon$ $PN_N$
    b.Compute the Corresponding $LFT_N$=$EST_N$ + $D_N$
    c.Label the node N as ($EST_N$,$LFT_N$)
    d.ULN=ULN - {($EST_N$,$LFT_N$)}
    e.LN = LN + {($EST_N$,$LFT_N$)}
end
//no unlabeled nodes remain.
3.End.
```

Fig. 4. Critical Path Algorithm

The assignment of tasks to employees are given on the devotion basis. We have choice to do the division of devotion in percentage. The gene value of chromosome is an integer in the range 0 to 8. That is, we have set of devotions in gene values with percentage considered as per following equation. If employee E_i is assigned tasks T_j , and gene value is "1" then his or her devotion is 12.5 % of his(r) capacity . The capacity is property of Full time Software Professional (FSP) and its unit may be hours/day, days/week, hours/week. e.g. an employee can give 10 hours "means" :if employee works with capacity of 40 hours/week and his devotion is 25%. The devotion of employee can be given as

$$Devotion = \frac{g}{G_{max}} \times 100 \tag{14}$$

where g is gene value and G_{max} is max value of gene. We have used string representation of chromosome in our paper as an employee task assignment schedule. Each chromosome shows the schedule of assignment of skilled employees and tasks with specified and derived time of the software project.

6.4 Solution Representation in our approach

We proceed to describe the elements of a solution for the problem. A solution can be represented with a matrix AGSHD =(x_{ij}) of size E x T, where $x_{ij} \varepsilon$ [0,1]. The value x_{ij} represents the fraction of the working time that the employee E_i dedicates to the task T_j in terms of months. A sample example of a problem solution is given in Table II, where a software project with 7 tasks is performed by a team of 5 employees. '1' indicates 100%.

Table 6. a Sample example of solution in terms of Schedule representation.

	t_0	t_1	t_2	t_3	t_5	t_6
e_0	0.125	0.625	1.0	0.375	0.125	0.625
e_1	0.625	0.25	0.625	0.125	1.0	0.125
e_2	0.375	1.0	0.125	1.0	0.625	0.625
e_3	1	0.625	0.375	0.375	0.375	0.375

7. OPERATIONS, MATHEMATICAL MODELING AND ADAPTIVE FITNESS [?]

7.1 Modified objective function

Modified objective function is defined as

$$\chi(SHD_s) = F(SHD_s) \times (1 + DP_g) \tag{15}$$

$$\overbrace{DP_g = \frac{(P_{max} + P_s)}{P_{max} - P_{ave}}}^{if \quad P_s \geq P_{ave}} \tag{16}$$

$$\overbrace{DP_g = \frac{(P_{ave} + P_s)}{P_{ave} - P_{min}}}^{if \quad P_s < P_{ave}} \tag{17}$$

$$\overbrace{DP_g = 0}^{if \quad P_s \doteq 0} \tag{18}$$

In this way, the penalty function is kept free from any pre-defined or user defined constants, and the degree of penalty can vary according to the level of violation instead of being constant during the schedule process. The outlines of the adaptive mutation and crossover operators used in the current work, based on the methods suggested by Srivinas and Patnaik (1994) but modified and described in next subsection. P_m and P_c represent a number that demonstrate the task duration shifted by mutation in the individual and exchanged by crossover between the pairs. In the following formulations, the probability of mutation and cross-over depend on the fitness value of the solutions, and vary according to the fitness value. Therefore, the user is free from defining any value for those [53].

7.2 Fitness for AGA

The calculation and flow of the fitness for our model is sequenced in following manner. **s^{th} number in every equation indicates schedule number or chromosome number in generation** Fitness of chromosome for s^{th} schedule is given by

$$FC_s = \frac{1}{ND_s + P_s} \tag{19}$$

,where P_s= Total Penalty of s^{th} schedule,

$$ND_s = SHD_s/AD_s \tag{20}$$

where, ND is Normalised duration,AD is Average Duration for s^{th} schedule.

7.3 Project Duration (Schedule)

An each completed task is checked according to the TPG. For each individual, the sequence and the task completion is checked and penalty is given if the task is not completed. The constant penalty and reward technique is adapted to get good individuals, instead of making it invalid,completely. The project duration is calculated by

$$AGSHD_s = \sum_{i=1}^{e}\sum_{j=1}^{n} DV_{(E_i,T_j)} \tag{21}$$

where,

$$DV_{(E_i,T_j)} = devotion\, of\, employee\, E_i\, to\, Task\, T_j\, in\, months. \tag{22}$$

7.3.1 **Project (Schedule) Cost**. The total schedule cost (SC) or project cost (PC) is obtained by summation of multiplication of devotion of each employee with each task and salary of each employee per month.

$$SC_s = \sum_{i=1}^{e}\sum_{j=1}^{n} DV_{(E_i,T_j)} \times SE_{E_i} \tag{23}$$

Maximum cost of the project is calculated by,

$$MSC_{max} = \sum_{i=1}^{n} CPSHD_s \times SE_{max} \tag{24}$$

where, SE_{max} is maximum salary of employee among all employees.

7.4 Total Schedule penalty

Total penalty [32] is addition of penalties regarding time,individual task and head count. There are some competing objectives which may give the delay to get the right solution. We have example of making equal distribution of employees in project which is exactly opposite to the head count constrain. Making trade off in the contradictory objectives is must and common in project management [5]. The solution is to make the one of them hard and other one soft or make the both constrained soft. The total schedule penalty is calculated and given by

$$P_s = NHCP_s + NTP_s + TNITP_s \tag{25}$$

where, $NHCP_s$, NTP_s, $TNITP_s$ are normalised head count, total time and task incompleteness penalty respectively.

7.4.1 Head count penalty (team size penalty). The head count of each task is the number of employees assigned to that task. The effort is dependent on the team size and the team productivity. Effort changes due to change in the number of employee in the project. We calculated the team size for current schedule and compare the team size calculated initially by Eqn-1. The difference between them is taken as team size penalty. The following constraint is taken as head count constraint for every individual task, Hence, the hard constraint is, HC of each task should be ± 1 of Head count calculated by COCOMO.

$$HCP_{T_i} = \begin{cases} 0 & if Condition 1 \\ |CHC_{T_i} \pm 1 - AGHC_{T_i}| & Otherwise \end{cases} \quad (26)$$

$$Condition1 \equiv CHC_{T_i} \pm 1 \doteq GHC_{T_i} \quad (27)$$

$$NHCP_s = \frac{\sum_{i=1}^{n} HCP_{T_i}}{TCHC_s} \quad (28)$$

where TCHC$_s$ is total COCOMO head count of sth schedule.

7.4.2 Threshold penalty (Time penalty) . The threshold penalty is calculated by taking the difference between AGSHD and CPSHD.

$$TP_s = \begin{cases} rate * (CPSHD_s - AGSHD_s) & if\ Condition1 \\ 1 \times 10^9 & if\ Condition2 \end{cases} \quad (29)$$

$$Condition1 \equiv COSHD_s <= AGSHD_s <= CPSHD_s \quad (30)$$

$$Condition2 \equiv GSHD_s > CPSHD_s \quad (31)$$

Where,rate is penalty per unit time (We kept rate= 1 penalty/months), AGSHD is schedule obtained by our genetic approach, CPSHD is COCOMO pessimistic schedule and COSHD is COCOMO optimistic schedule. **In SPM, schedule is also called as project duration or DUR or DU.**

$$NTP_s = \frac{TP_s}{CPSHD_s} \quad (32)$$

NTP$_s$ is Normalised time penalty.

7.4.3 Penalty For Individual Incomplete Task [9]. The penalty of individual task is called incompleteness. This incompleteness is duration difference between optimistic duration and GA obtained duration of the task.

$$\sum_{i=1}^{n} NITP_{T_i} = \sum_{i=1}^{n} (AGOD_{T_i} - COD_{T_i})/GPD_{T_i}. \quad (33)$$

$$TNITP_s = \sum{}^{n}{}_{i=1} NITP_{Ti} \quad (34)$$

The incompleteness is normalised by above equation.

7.5 Adaptive Mutation

Mutation is done to change the direction search space as lowest fitness values continuation doesnt provide good solution. Mutation rate is changed according (to Togans [53] following formula) provided the chromosome has worst fitness in the pool.

$$P_m = \begin{cases} 0.5(f_{max} - f)/(f_{max} - f_{ave}) & f \geq f_{ave}. \\ (f_{ave} - f)/(f_{ave} - f_{min}) & f < f_{ave}. \end{cases}$$

Here, f is the fitness of an individual, f_{ave} is the average fitness value of the population, f_{max} and f_{min} are the maximum and minimum fitness value of an individual in the population respectively. Pm is mutation rate.

We have done simply the summation of all the objective function and calculate the fitness value. Every objective is given same preference. The weight is equally distributed in the objectives itself. The proposed SGA with adaptive approach for modified fitness function given by Togan, Patnaik is considered to get the more suitable fittest chromosome from adaptive approach [53].

7.6 Adaptive Crossover

The crossover operator mimics the way in which bisexual reproduction passes along each parents good genes to the next generation [18]. Two parent create two new offsprings by combining their genes typically according to following pseudo code. Crossover uses both inheritance and variation to improve the performance of the population while retaining its diversity of population [8]. In our approach, following flow of operation of crossover is experimented. Pseudocode for the same is given bellow.

7.6.1 Adaptive Elitist Crossover (EX). where:
P(t) is equal to P$_c$; Comments ? The selection process in SGA is always preceded by the crossover process. But ,in the EX above method, both processes are integrated. The entire population is randomly shuffled during the first step. Then,two new vectors are created by crossover from each successive pair of parental vectors. Two best vectors are singled out and taken as offspring to the next population. ? Some times premature convergence may be due to the reason of traditional way of elitist selection application on the level of the entire population. So, we can apply this an EX elitist selection on the family level.

$$P_c = \begin{cases} (f_{max} - f')/(f_{max} - f_{ave}) & f' \geq f_{ave}. \\ 0 & f' < f_{ave}. \end{cases}$$

Where, f' is the larger of the fitness value of the solutions to be crossed [53]

7.7 Selection :Elitist strategy

Elitist strategy is utilized for avoiding destroying the best individual per generation. Specifically described as follows: If the next generation of groups of individual fitness value is less than the current population of individual fitness value, the best individuals in the current groups or adaptation value is greater than the value of the next generation of the best individual fitness multiple individuals directly copied to the next generation. The elitist strategy ensures that the current best individual will not be destructed by crossover and mutation operations [43]. We used separate ádaptive crossover and elitist selection, Elitist adaptive crossover.

7.7.1 Stopping Criterion. Though, it is, usually, expected that solution quality improves with the additional generations in GA. Typically, genetic algorithms terminate after a predetermined number of generations passed or after a sequence of consecutive generations without objective function improvement. Alternatively, the algorithm can terminate after the population is sufficiently homogenized, as measured by objective function variance [54]. We are interested in the temporal (generational) performance in elitism selection strategy [42]. we opted to utilize a maximum generation as a stopping criterion..

8. INPUT OUTPUT TABLES

We took the input as configuration files which have already been given by Albas and paper written by Xing,Tang and Ting. We have written C program to get the output as employees properties, task properties and TPGs. The outputs in terms of all these properties has been given as the parameters and properties in INPUT Table-7 to 9. In tern, These parameters are taken as inputs to our implementation. The table-10 onward all tables represents outputs of our approach.

Table 7. INPUT:Configuration file showing mathematical notations for T10E5S4 as an example [56]

task.6.skill.number=3	task.6.cost=12.0
task.3.skill.2=2	task.0.cost=4.0
task.3.skill.1=5	task.9.skill.number=2
task.4.cost=7.0	employee.number=5
task.3.skill.0=3	employee.3.skill.number=5
employee.3.skill.4=5	task.8.skill.1=5
employee.3.skill.3=1	task.8.skill.0=2
employee.3.skill.2=6	task.4.skill.1=3
employee.3.skill.1=3	task.4.skill.0=6
employee.3.skill.0=0	task.3.skill.number=3
graph.arc.9=1 7	employee.4.skill.3=2
graph.arc.8=5 6	employee.4.skill.2=3
graph.arc.7=4 5	employee.4.skill.1=5
graph.arc.6=3 5	employee.4.skill.0=7
graph.arc.5=0 5	employee.2.skill.2=8
graph.arc.4=3 4	employee.2.skill.1=9
graph.arc.3=2 4	employee.2.skill.0=3
graph.arc.2=0 4	employee.0.skill.3=0
graph.arc.1=1 3	employee.0.skill.2=8
task.1.skill.0=0	task.2.cost=7.0
task.5.cost=8.0	employee.3.salary=9501.80
employee.1.skill.3=1	employee.0.skill.1=3
employee.1.skill.2=8	task.2.skill.number=2
employee.2.salary=9935.60	employee.0.skill.0=1
graph.arc.15=6 8	task.7.skill.number=2
graph.arc.14=5 8	task.9.cost=5.0
graph.arc.13=4 8	task.1.skill.number=3
graph.arc.12=2 8	task.9.skill.1=0
graph.arc.11=4 7	task.9.skill.0=2
graph.arc.10=3 7	employee.0.skill.number=4
task.4.skill.number=3	

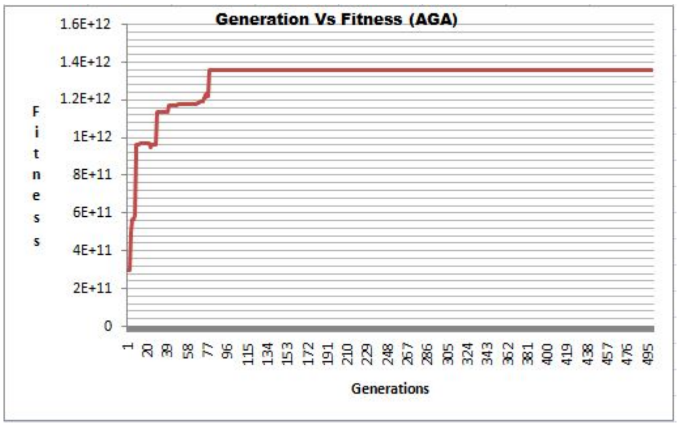


Fig. 5. Generation Vs Fitness T20E5S4

Table 8. INPUT : Task Properties for T20E5S4.

Task ID	Effort (PM)	CD (Mths)	CPD (Mths)	CHC	Skills Required
Task0	11	6.5	10.54	3.3	1,5,9
Task1	3	4.3	6.88	1.3	3,6,9
Task2	12	6.7	10.84	3.5	2,6
Task3	8	5.9	9.49	2.6	0,8,9
Task4	12	6.7	10.84	3.5	0,5,7
Task5	7	5.6	9.08	2.4	1,7
Task6	15	7.2	11.67	4.0	0,6,8
Task7	21	8.1	13.03	5.1	4,5
Task8	11	6.5	10.54	3.3	1,3
Task9	18	7.7	12.39	4.6	5,7,9
Task10	10	6.3	10.22	3.1	1,3,5
Task11	8	5.9	9.49	2.6	0,1,2
Task12	17	7.5	12.16	4.4	6,7,8
Task13	15	7.2	11.67	4.0	0,5,9
Task14	16	7.4	11.92	4.2	0,1,7
Task15	7	5.6	9.09	2.4	3,7
Task16	10	6.3	10.22	3.1	4,6,8
Task17	10	6.3	10.22	3.1	0,1
Task18	11	6.5	10.54	3.3	0,1,5
Task19	15	7.2	11.67	4.0	6,8

Table 9. INPUT : Employee properties for T20E5S4.

Emp Id	Salary	Months	work load factor	Skills
0	9793	3	1	0,1,5,7,9
1	9545	3	1	0,1,6,8,9
2	10131	3	1	1,2,4,8
3	10252	3	1	0,5,6,9
4	10944	3	1	0,3,7,8

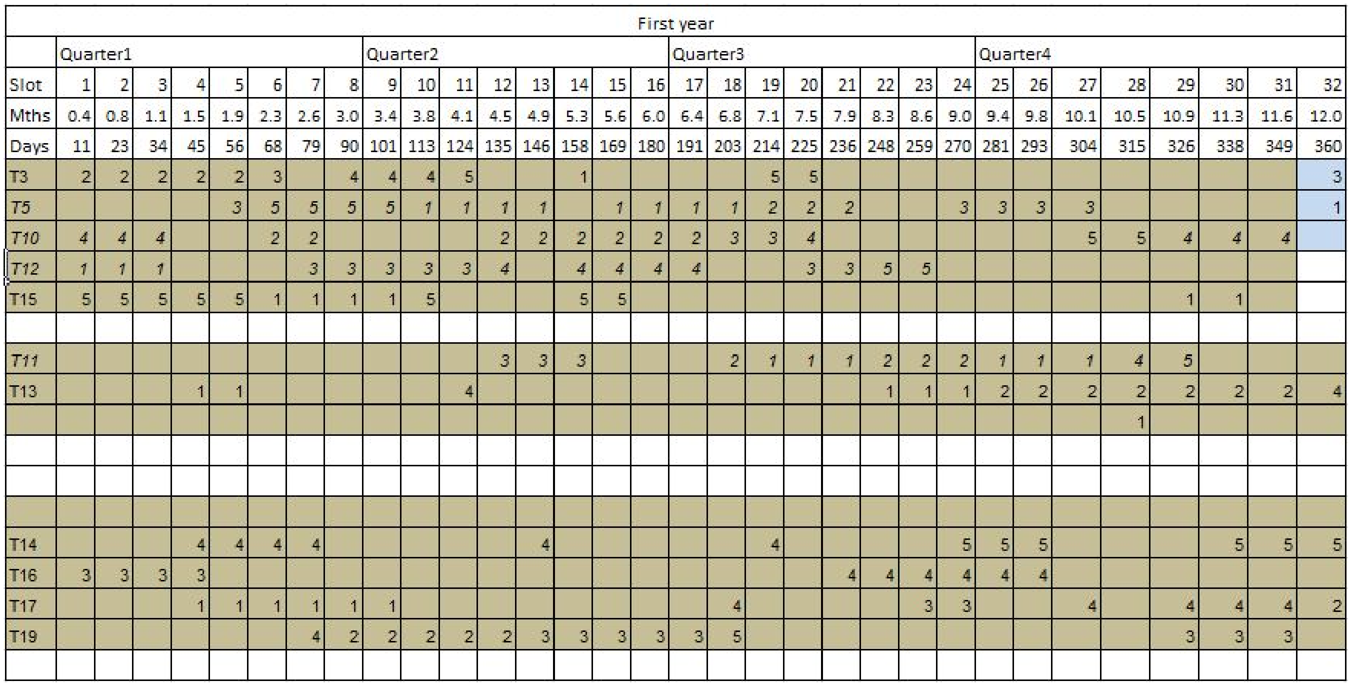
Table 10. OUTPUT:Employee working times for T20E5S4.

Employee	Time	No. of Task Per Employee
0	25.875	15
1	22.025	15
2	27.975	14
3	24.4	15
4	21.525	16

9. RESULT AND DISCUSSION

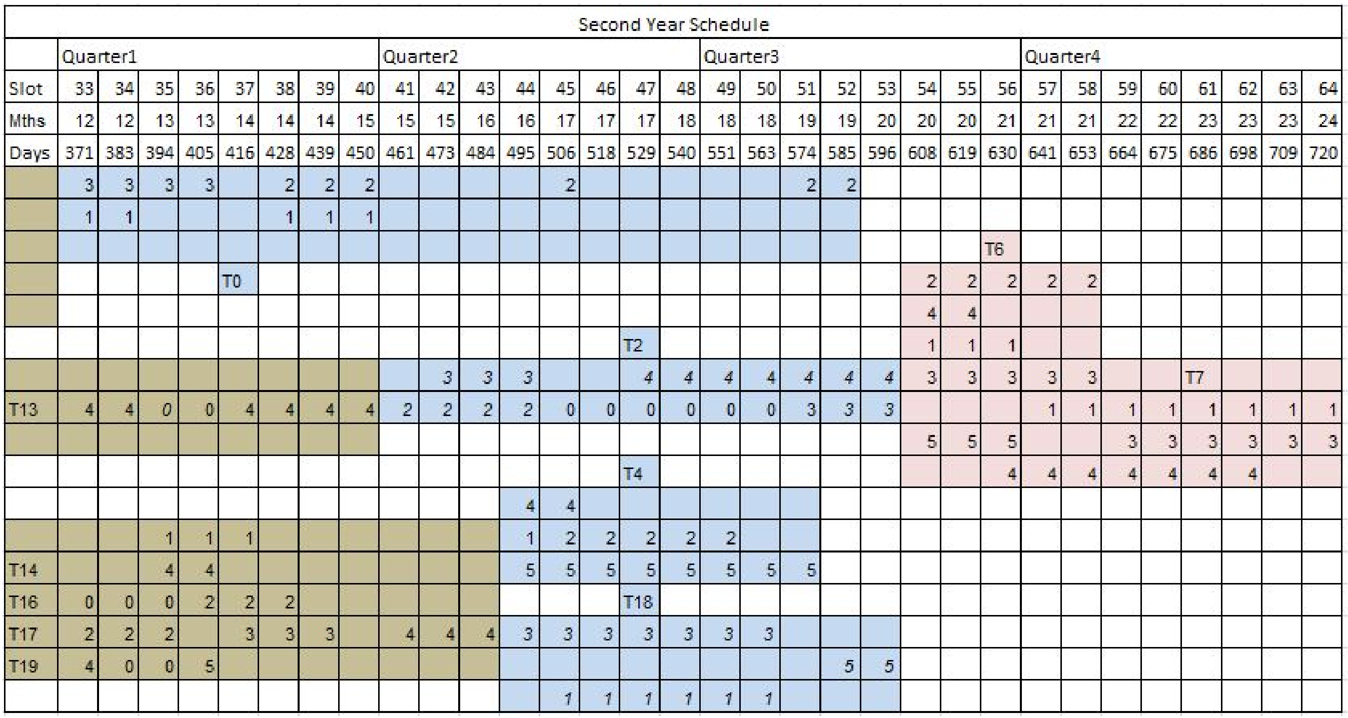
9.1 Analysis and Discussion

The java code is written in netbean 7.1 IDE for JDK 1.6 with different java classes available for GA. The average number of computation or evaluations needed to reach maximum of AGA is in Table XIII. In GAs,P$_c$ and P$_m$ are usually assigned constant value but in our approach we made crossover and mutation as variable of fitness function which produces more efficient search. We are indirectly mutating LSB for high-fitters thus it improves the accuracy. AGA intends put low-fitters in mutation to play role in evolution.
In order to solve, the feeble adaptability and the imbalance between random search and local search in the SPSP, a new adaptive genetic algorithm (AGA) is presented in this paper. The superiority of this algorithm was the adaptation achieved by adjusting the crossover



First year																																
	Quarter1								Quarter2								Quarter3								Quarter4							
Slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Mths	0.4	0.8	1.1	1.5	1.9	2.3	2.6	3.0	3.4	3.8	4.1	4.5	4.9	5.3	5.6	6.0	6.4	6.8	7.1	7.5	7.9	8.3	8.6	9.0	9.4	9.8	10.1	10.5	10.9	11.3	11.6	12.0
Days	11	23	34	45	56	68	79	90	101	113	124	135	146	158	169	180	191	203	214	225	236	248	259	270	281	293	304	315	326	338	349	360
T3	2	2	2	2	2	3		4	4	4	5			1					5	5												3
T5					3	5	5	5	5	1	1	1	1		1	1	1	1	2	2	2			3	3	3	3					1
T10	4	4	4			2	2					2	2	2	2	2	2	3	3	4							5	5	4	4	4	
T12	1	1	1				3	3	3	3	3	4		4	4	4	4			3	3	5	5									
T15	5	5	5	5	5	1	1	1	1	5				5	5														1	1		
T11												3	3	3				2	1	1	1	2	2	2	1	1	1	4	5			
T13				1	1						4											1	1	1	2	2	2	2	2	2	2	4
																												1				
T14				4	4	4	4						4						4					5	5	5				5	5	5
T16	3	3	3	3																	4	4	4	4	4	4						
T17				1	1	1	1	1	1									4					3	3			4		4	4	4	2
T19							4	2	2	2	2	2	3	3	3	3	3	5											3	3	3	

Fig. 8. Schedule for first year for Task Precedence Graph for T20E5S4



Second Year Schedule																																
	Quarter1								Quarter2								Quarter3								Quarter4							
Slot	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
Mths	12	12	13	13	14	14	14	15	15	15	16	16	17	17	17	18	18	18	19	19	20	20	20	21	21	21	22	22	23	23	23	24
Days	371	383	394	405	416	428	439	450	461	473	484	495	506	518	529	540	551	563	574	585	596	608	619	630	641	653	664	675	686	698	709	720
	3	3	3	3		2	2	2					2						2	2												
	1	1				1	1	1																								
																								T6								
					T0																	2	2	2	2	2						
																						4	4									
															T2							1	1	1								
										3	3	3			4	4	4	4	4	4	4	3	3	3	3	3			T7			
T13	4	4	0	0	4	4	4	4	2	2	2	2	0	0	0	0	0	0	3	3	3				1	1	1	1	1	1	1	1
																						5	5	5			3	3	3	3	3	3
															T4									4	4	4	4	4	4	4		
												4	4																			
			1	1	1							1	2	2	2	2	2															
T14			4	4								5	5	5	5	5	5	5	5													
T16	0	0	0	2	2	2									T18																	
T17	2	2	2		3	3	3		4	4	4	3	3	3	3	3	3	3														
T19	4	0	0	5																5	5											
													1	1	1	1	1	1														

Fig. 9. Schedule for second year for T20E5S4

rate and mutation rate. At the same time, the search property has been balanced by restricting crossover and mutation. To insure the best chromosome pass to the next generation, we immediately reserved the best chromosome. The developed algorithm had been tested by benchmark problems. Computational results show this adaptive genetic algorithm (AGA) has an effective search behavior.

Third Year Schedule																						
	Quarter1								Quarter2								Quarter3					
Slot	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86
Mths	24.4	24.8	25.1	25.5	25.9	26.3	26.6	27.0	27.4	27.8	28.1	28.5	28.9	29.3	29.6	30.0	30.4	30.8	31.1	31.5	31.9	32.3
Days	731	743	754	765	776	788	799	810	821	833	844	855	866	878	889	900	911	923	934	945	956	968
											T1											
								5	5				0	0	0							
								2	2	2	2											
								1	1	1		0	0	0	0							
	0			T9				4	4	4	4	4	4	4	4				T8			
		5	5	5	5	5										2	2	2	2	2	4	
		4	4	4	4											1	1					
		1	1	0	0	0										3	3	3	3	3	3	3
	3	2	2	2	2	2	2									5	5	5	5	5	5	0

Fig. 10. Schedule for Third year for T20E5S4

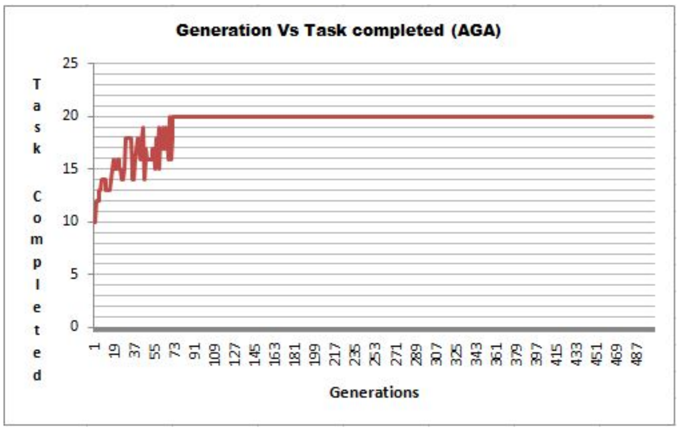


Fig. 6. Generations Vs Task Completed T20E5S4

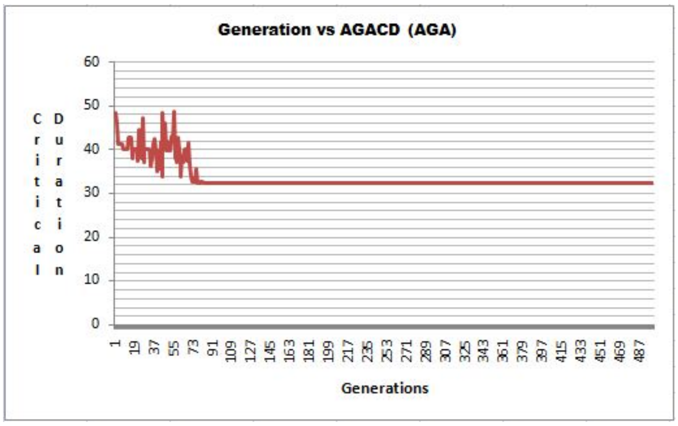


Fig. 7. Generation Vs Critical Duration for T20E5S4

This can get away from local optimal and avoid premature convergence. Also the convergence speed increases [19]
We have taken approach elitist selection. Though the AGA plays indirect role of elitist approach. We, also, took various penalties and reward to the parameters of the objective functions .Experiencing the various ways of the execution gives the interesting conclusions. For a quality project, task must be completed within time, between pessimistic optimistic duration, along with satisfying nearly all soft and hard constraints. In our problem, we considered the effort, duration, head count defined by Bohem for making correction in the fitness value of individual schedule. We considered

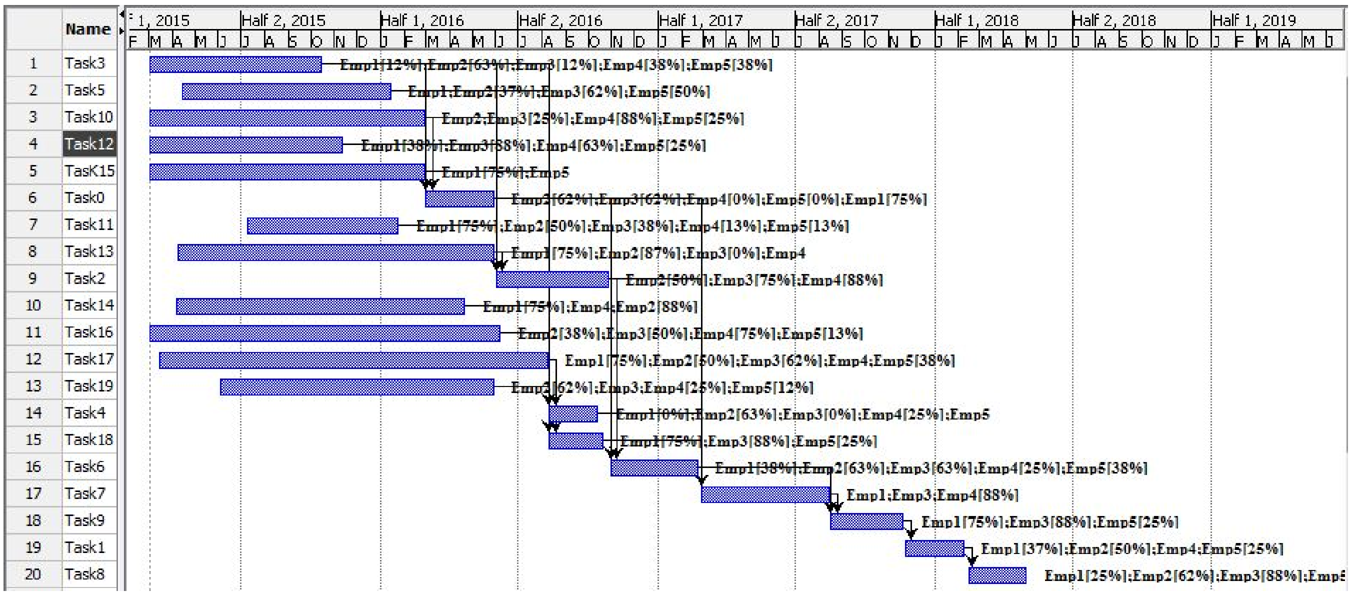


Fig. 11. Graph shows assignment of employee, duration for each employee to tasks, critical path duration, total duration for T20E5S4

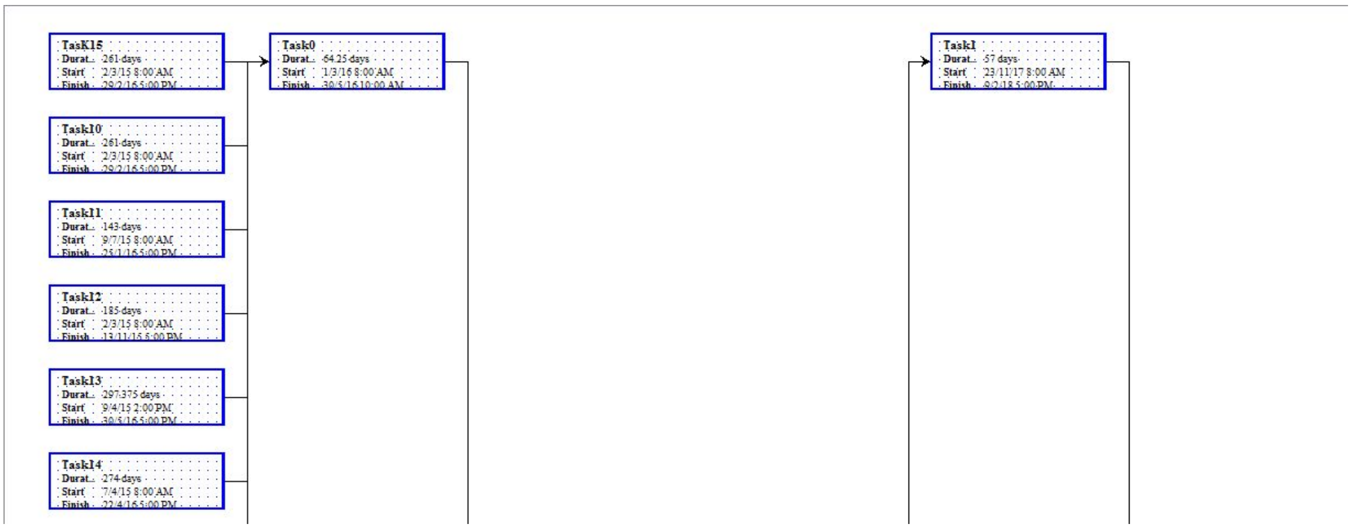

TasK15 Durat... 261 days Start 2/3/15 8:00 AM Finish 29/2/16 5:00 PM
Task10 Durat... 261 days Start 2/3/15 8:00 AM Finish 29/2/16 5:00 PM
Task11 Durat... 143 days Start 9/7/15 8:00 AM Finish 25/1/16 5:00 PM
Task12 Durat... 185 days Start 2/3/15 8:00 AM Finish 13/11/15 5:00 PM
Task13 Durat... 297.375 days Start 9/4/15 2:00 PM Finish 30/5/16 5:00 PM
Task14 Durat... 274 days Start 7/4/15 8:00 AM Finish 22/4/16 5:00 PM
Task0 Durat... 64.25 days Start 1/3/16 8:00 AM Finish 30/5/16 10:00 AM
Task1 Durat... 57 days Start 23/11/17 8:00 AM Finish 9/2/18 5:00 PM

Fig. 12. First part of Network diagram for T20E5S4

COCOMO-II effort estimation, schedule calculation, Head count for our paper and implementation. In preliminary run, we used the right extinctive selection for 2 to 5 individuals (having less fitness) for having zero production rate for some generation to discard the some but little number of chromosomes which dont tends to give super-individuals. We used elitist selection scheme which enforces to go through selection along with their parent by making some individual duplicates by constrained wise changes in the chromosomes.Comparison Table of simple GA(Ting [56]),AGA by our approach and ACO (Ting [56]) shows that AGA has good result than others. Figure 2 and onwards gives the various graphs and chart. These graphs gives the behaviour of AGA with parallelism. In parallelism ,if some skilled employees is idle then we allocate that task to the employee by making allocation of randomized genes. The execution of sequenced task are considered as per the principles of parallelism of project management.We see if all tasks are completed first within group of sequenced parallel tasks, in turn, we can start the next sequence of task immediately after the completion of all the tasks in the previous sequence. All predecessors of that task must be completed before going to execute successor tasks or follower tasks. Equal distribution of the work employee is considered for the sake of making equilibrium in the division of a task but it is exactly opposite to the head count of the task required.

9.2 Computational AGA results

The results for the sample example case for the given 20 task, 05 employee and 10 skills are shown in all the tables except comparative table IX. The cost and time decreases as we do progress from

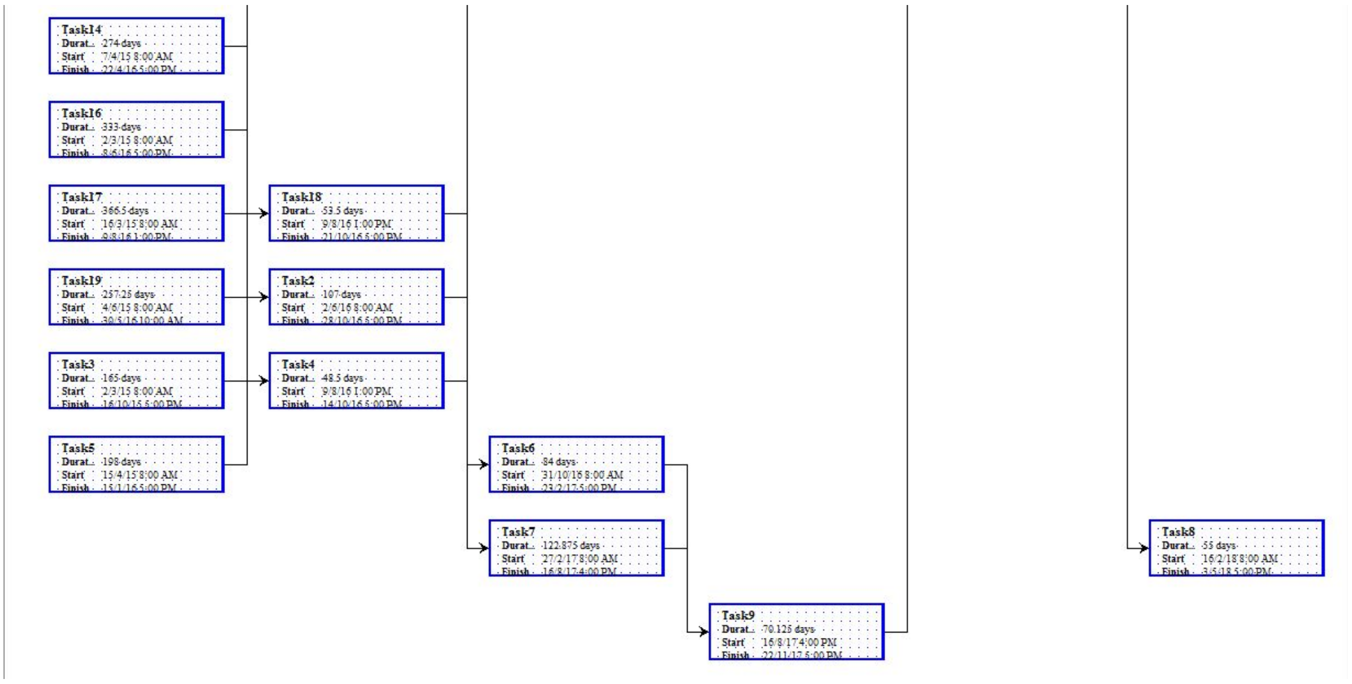


Fig. 13. Second part of Network diagram for T20E5S4

Table 11. COCOMO and AGA durations for T20E5S4 Where, $\mathrm{COD}_{T_i} \leq \mathrm{AGAD}_{T_i} \leq \mathrm{CPD}_{T_i}$.

	CD	COD	AGAD	CPD
Task0	6.59	5.66	6.00	10.54
Task1	4.30	3.70	6.38	6.88
Task2	6.78	5.83	6.38	10.84
Task3	5.93	5.10	4.88	9.49
Task4	6.78	5.83	5.63	10.84
Task5	5.68	4.88	7.50	9.09
Task6	7.29	6.27	6.75	11.67
Task7	8.14	7.00	8.63	13.03
Task8	6.59	5.66	7.50	10.54
Task9	7.74	6.66	6.38	12.39
Task10	6.38	5.49	7.13	10.22
Task11	5.93	5.10	5.63	9.49
Task12	7.60	6.53	6.38	12.16
Task13	7.29	6.27	7.88	11.67
Task14	7.45	6.41	6.38	11.92
Task15	5.68	4.88	5.25	9.09
Task16	6.38	5.49	5.25	10.22
Task17	6.38	5.49	9.75	10.22
Task18	6.59	5.66	5.63	10.54
Task19	7.29	6.27	6.00	11.67

Table 12. OUTPUT: Tasks duration in Months with employee for T20E5S4.

	Emp0	1	2	3	4	AGAD
Task0	2.25	1.875	1.875	0	0	6
Task1	1.125	1.5	0	3	0.75	6.375
Task2	0	1.5	2.25	2.625	0	6.375
Task3	0.375	1.875	0.375	1.125	1.125	4.875
Task4	0	1.875	0	0.75	3	5.625
Task5	3	1.125	1.875	0	1.5	7.5
Task6	1.125	1.875	1.875	0.75	1.125	6.75
Task7	3	0	3	2.625	0	8.625
Task8	0.75	1.875	2.625	0	2.25	7.5
Task9	0.75	2.25	0	1.5	1.875	6.375
Task10	0	3	0.75	2.625	0.75	7.125
Task11	2.25	1.5	1.125	0.375	0.375	5.625
Task12	1.125	0	2.625	1.875	0.75	6.375
Task13	2.25	2.625	0	3	0	7.875
Task14	1.125	0	0	3	2.25	6.375
Task15	2.25	0	0	0	3	5.25
Task16	0	1.125	1.5	2.25	0.375	5.25
Task17	2.25	1.5	1.875	3	1.125	9.75
Task18	2.25	0	2.625	0	0.75	5.625
Task19	0	1.875	3	0.75	0.375	6

one generation to another generation. The fitness value takes a constant path after some generation as it increases, it takes constant value. Though, the fitness gives constant value but it gives some different solutions by considering all the composite components of the schedule. We studied combination of different crossover types with adaptive P_c,P_m. The generation is set to 500 for this specific problem of 20 Tasks, 5 Employees, 10 Skills. We got the following as outputs

—schedule as chromosome in terms of Employee- Task 2D matrix

—conversion of these matrix in terms of months assigned as per the devotion for same task, assigning of the task as per the TPG and sequence

Table 13. Xover Vs Avg Computatin time for T20E5S4.

Xover Methodology	Avg (SGA) Computation	Avg (AGA) Computation
One point	250	90
Two pont	210	75
Uniform	290	100
Selective	360	120

—addition of each gene to get the value of time required for tasks, employees working time
—getting the distribution of employee over the number of tasks
—team size for completing the each task
—concurrent parallel adjustment of each employee with the task assigned to skilled employee
—time required to complete the project i.e. schedule
—Cost of the schedule
—fitness of the schedule.

The figures 2 to 10 shows the above things in the chart. The cost and time decreases as we do progress from one generation to another generation. The fitness value takes a constant path after some generation as it increases, it takes constant values. Though the fitness gives constant value but it gives some different solutions by considering all the composite components of the schedule. We put crossover rate 0.9, mutation rate 0.01 initially and then the effect of adaptive approach is observed which makes P_m and P_m changes according to the fitness from generation to generation as stopping criteria is 500 number of generations as different type of crossover requires different computation time.

9.3 Gantt Chart of sample example as an representation of task, breakdown

The schedule created is shown in Gantt Chart which shows tasks assigned to skilled employees, critical path, dependency between tasks as per input TPG, Starting and ending days and duration of each tasks. It shows the sequence and flow of the tasks with starting and ending point with employee allocations in slots. Fig.6 shows the graph which shows same information as above for schedule of 20 Tasks, 5 employees, 10 skills example. It shows the concept of SS, SE, ES, EE examples also and can be used for showing it.

9.4 Comparative discussion

Here, we compared the results of our model's with Ting,Tang's [56] results. Somewhere, it has been found that the result of ACO (Ant colony optimization) are better and some time our results are. This observation can also be seen in the results of Ting,Tang. ACO is better than GA for some of the smaller problems and our approach GA put behind the ACO and the result of Ting and Tang. Our results are good when problem becomes complicated and complex due to GA's global optimization nature, due to combination of crossover types and changes in the mutation rate according to fitness of individual. Our results produced may give hope of making project manager feel satisfy about the schedule.

10. CONCLUSION

Not only SGA or GA but also AGA gives various chance to us to take tour of various parameters variability to get the optimistic solution. All operators like adaptive crossover, adaptive mutation, adaptive fitness gives different combinations of the solution. It is the god gifted mechanism that gives chance of using constant as well as mathematical modelled penalties to get the best optimal solution. The constant penalty approach gives some what less result than adaptive approach in the penalty.
In preliminary run, we set the pool of chromosome in such a way that, after preliminary run the chromosomes in the pool get some combinations of the super individual, best individual, normal individuals and zero product individual so that the natural test of GA is kept on. In preliminary run, we got some good solution in chromosome pool to propagate to the next generation. Figure 3 to 7 gives different graphs .These graphs gives the behaviour of AGA with parallelism in scheduling [30]. In parallelism ,if some skilled employees is idle then we allocate that task to the employee by making allocation of randomized genes. The execution of sequenced tasks are considered as per the principles of parallelism of project management [15] .We see, if all tasks are completed first within group of sequenced parallel tasks, in turn, we can start the next sequence of task immediately after the completion of all the tasks in the previous sequence. All predecessors of that task must be completed before going to execute successor tasks or follower tasks. Equal distribution of the work employee is considered for the sake of making equilibrium in the division of a task but it is exactly opposite to the head count of the task required .

Table 14. OUTPUT:Comparative chart of our and Ting-Tang approach for G2 Group **Employee possesses 4-5 skills.**

Group	Instance	Algorithm	Duration
G2	5e10t	ACS-SPSP	22.94651
		GA	23.63111
		AGA	20.875
	10e10t	ACS-SPSP	14.21091
		GA	16.28071
		AGA	16.125
	15e10t	ACS-SPSP	8.032931
		GA	8.2231
		AGA	8.125
	20e10t	ACS-SPSP	6.306531
		GA	6.01912
		AGA	6.00

Table 15. OUTPUT:Comparative chart of our and Ting-Tang approach for G3 Group.**Employee possesses 6-7 skills**

Group	Instance	Algorithm	Duration
G3	5e10t	ACS-SPSP	22.11891
		GA	23.15251
		AGA	23.625
	10e10t	ACS-SPSP	14.20831
		GA	13.2522
		AGA	11.625
	15e10t	ACS-SPSP	8.2368
		GA	8.05221
		AGA	7.75
	20e10t	ACS-SPSP	6.020961
		GA	6.284971
		AGA	6.125

Table 16. OUTPUT:Comparative chart of our and Ting-Tang approach for G4 group.**Employee possesses 2-3 skills**

Group	Instance	Algorithm	Duration
G4	5e10t	ACS-SPSP	23.6223
		GA	24.26261
		AGA	23.125
	10e10t	ACS-SPSP	14.1833
		GA	15.23891
		AGA	14.5
	15e10t	ACS-SPSP	8.8731
		GA	8.89696
		AGA	7.625
	20e10t	ACS-SPSP	6.526491
		GA	7.055161
		AGA	6.875

11. FUTURE WORK

In the future, we can use the adaptive elitist-population search method, a new technique for evolving parallel elitist individuals for multimodal function optimization. The technique is based on the concept of adaptively adjusting the population size according to the individuals dissimilarity and the elitist genetic operators. The adaptive elitist-population search technique can be implemented with any combinations of standard genetic operators. To use it, we just need to introduce one additional control parameter, the distance threshold, and the population size is adaptively adjusted according to the number of multiple optima.
We have a plan to apply this technique to hard multimodal engineering design problems with the expectation of discovering novel solutions. We will also need to investigate the behavior of the AEGA on the more theoretical side. The performance of the AEGA can be compared against the fitness sharing, determining crowing and clonal selection algorithms.
Second issue might be Running huge parallel project which is tough task for the PM. Future work may give us to do the parallel among the module to module or project to project. Multi-project scheduling problem is challenging issue using various type of genetic algorithm.
Another angle and the area for Human resource management in software is to calculate the efficiency of each employee for getting better capacity to do the work in that particular area. It has been observed that human being do the work fast and keep the tempo of his work when he likes to do the work in his domain and area. The psychological measurement and other physiological aspects are directly proportional to eagerness and enthusiastic property of human being which is very important factor for increasing the productivity of the employee and its team. In this paper, adaptive approaches are proposed for both the penalty function, and crossover and mutation probabilities in order to relieve the user from predicting any values needed prior the optimization, and enhance the performance of the GA in optimizing structural systems. A strategy is also considered for member grouping to reduce the size of the problem and the number of iteration to reach the optimum solution. From the results, it is possible to say that the adaptive improvements presented in the current study are effective and enhance the performance of the GA. It can be concluded that member grouping proposed in the future work together with the adaptive approaches improve the capability of GA to catch the global optimum.

12. REFERENCES

[1] D.B. Hanchate A. Thorat, R. H. Ambole. review on multi-mode resrouce constrained project scheduling problem". *IJCSET*, 2012.

[2] T. Back. Varying the probability of mutation in genetic algorithms. In *Proc. Third Int. Con. Genetic Algorithms*, 1989.

[3] Charles F Baer. "does mutation rate depend on itself". *Genetics*, pages 295–304, 2008.

[4] Barry W. Boehm. Software engineering economics, 1981.

[5] Tao Zhang Carl K. Chang, Mark. J. Christensen. Genetic algorithms for project management. *annals of Software Engineering*, 11:107139, 2001.

[6] Carl K. Chang, Chikuang Chao, Thinh T. Nguyen, and Mark J. Christensen. Software project management net: A new methodology on software management. In *COMPSAC*, pages 534–539, 1998.

[7] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. London, 1859.

[8] Kalyanmoy Deb. Genetic algorithm in search and optimization: The technique and applications. In *Proc. of Int. Workshop on Soft Computing and Intelligent Systems*, pages 58–87, 1997.

[9] Rajankumar S. Bichkar Dinesh Bhagwan Hanchate. sps by combination of crossover types and changeable mutation sga. *IJCA*, pages 41–66, 2014.

[10] Charlesworth B Charlesworth D Crow JF Drake, JW. "rates of spontaneous mutation". *Genetics*, 1998.

[11] A. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2007.

[12] CarenMarzban AntonelloPasini Ellen Haupt, ValliappaLakshmanan and John K. Williams. Environmental science models and artificial intelligence. *Springer Science.*, 2009.

[13] CarenMarzban AntonelloPasini Ellen Haupt, ValliappaLakshmanan and John K. Williams. Environmental science models and artificial intelligence. *Springer Science.*, 2009.

[14] J. Francisco Chicano Enrique Alba. sofware project management with gas". *Information Sciences,SceinceDirect*, 2007.

[15] Davis E.W. and G.E. Heidorn. ian algorithm for optimal project scheduling under multiple resource constraints. *Management Science*, pages 41–66, 1971.

[16] T C. Fogarty. Varying the probability of mutation in genetic algorithms. In *Proc. Third Int. Con. Genetic Algorithms*, 1989.

[17] Futuyam. *The genetical theory of natural selection*. 2009.

[18] J. J. Grefenstette. Optimization of control parameters for genetic algorithms,. *IEEE Trans. Systems, Man, and Cybernetics*, 16:122–128, 1986.

[19] Ren-Wang Li Gui Yang, Yujun Lu. Adaptive genetic algorithms for the job-shop scheduling problems in hanintelligent control and automation. In *WCICA*, 2008.

[20] T Haynes. A comparison of random search versus genetic programming as engines for collective adaptation. In *In: Proc. of the ACM Symposium on Applied Computing*, 1997.

[21] Jurgen Hesser and Reinhard Manner. towards an optimal mutation probability for genetic algorithms.

[22] E. Horowitz and S. Sahani. *,Fundamentals of Computer Algorithms*. Galgotia Publications, 1999.

[23] Sam Hsiung and James Matthews. *Introduction to Genetic Algorithms*.

[24] Shengxiang Yang Imtiaz Korejo and ChangheLi. A comparative study of adaptive mutation operators for genetic algorithms. In *MIC 2009: The VIII Metaheuristics International Conference*, 2009.

[25] Kenneth Alan De Jongč. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan Ann Arbor, MI, USA, 1975.

[26] Bryant A. Julstrom. What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm. In *ICGA*, pages 81–87, 1995.

[27] PD Keightley. rates and fitness consequeces of new mutations in humans. *Genetics*, pages 295–304, 2012.

[28] Kosorukoff. Using incremental evaluation and adaptive choice of operators in a genetic algorithm,. In *Proceedings of the Genetic and Evolutionary Computation Conference*, 2002.

[29] Lam Law and K.Y.Szeto. Adaptive genetic algorithm with mutation and crossover matrices. In *IJCAI*, 2007.

[30] Keqin Li. scheduling parallel tasks on multiprocessor computers with efficient power management. *IEEE*, pages 41–66, 2010.

[31] Park Mark Anclif. "mutation rate threshold under chaning environments with sharp peak fitness function". *Journal of the Korean Physical Society*, 2008.

[32] Jeff D. Hamann Matthew P. Thompson and John Sessions. "selection and penalty strategies for genetic algorithms designed to solve spatial forest planning problems". *International Journal of Forestry Researc.*, page 14, 2009.

[33] Michalewich and D. Fogel. How to solve it ?: Modern heuristics. *Springer*, 2002.

[34] Z Michalewich. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer,, 1996.

[35] S. Forest Mitchell and J.H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In *First ECAL,*, 1992.

[36] S. Forest Mitchell and J.H. Holland. The royal road for genetic algorithms: Fitness landscapes and ga performance. In *First ECAL, 1992.*, 1992.

[37] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Companies, Inc., 1997.

[38] Z. Pan and L. Kang. An adaptive evolutionary algorithm for numerical optimization. In *Proc. of the 1st Asia-Pacific Conference on Simulated Evolution and Learning*, page 2734, 1996.

[39] Himanshu Bhalchandra Dave Parag Himanshu Dave. *Design and Analysis of Algorithms*. Pearson Education, 2008.

[40] Y. Attikiouzel R. J. Marks II D. B. Fogel Peter J. Angeline, M. Palaniswami and T. Fukuda (eds.). Adaptive and self-adaptive evolutionary computations. In *Computational Intelligence: A Dynamic Systems Perspective*, pages 152–163. IEEE Press, 1995.

[41] Michael Pinedo. *Scheduling:Theory, Algorithms, and Systems*. Kluwer Academic Publishers, 2013.

[42] and M. Kurttila. Pukkala. examining the performance of six heuristic optimisation techniques in different forest planning problems,. *Silva Fennica*, (2), 2005.

[43] SAN Ye Ren.Ziwu. improved adaptive genetic algorithm and its application research in parameter identification[j].. *Journal of System Simulation.*, pages 41–66, 2006.

[44] S. Sariel S. Uyar and G. Eryigit. A gene based adaptive mutation strategy for genetic algorithms. In *Proc. of the 2004 Genetic and Evolutionary Computation Conference*, 2004.

[45] A. Preece S. Yang, F. Coenen and A. Macintosh (eds.). Adaptive mutation using statistics mechansim for genetic algorithms,. In *Research and Development in Intellignet Systems XX.*

[46] J. Sarma and K. De Jong. Selection: generation gap methods. In *Handbook on Evolutionary Computation*, pages C2.7:1–C2.7:5. Institute of Physics Publishing and Oxford University Press, Bristol and New York, 1997.

[47] H. P. Schwefel. *Numerical Optimization of Computer Models.* Wiley, Chichester, 1981.

[48] J.E. Smith and T.C. Fogarty. Operator and parameter adaptation in genetic algorithms. *Soft Computing*, 1(2):81–87, 1997.

[49] R. S. Sutton and A. G.Barto. *, Reinforcement Learning An Introduction.* MIT Press, 1998.

[50] D. Thierens. Adaptive mutation using statistics mechansim for genetic algorithms,. In *Research and Development in Intellignet Systems XX.*

[51] D. Thierens. Selection schemes,elitist recombination and selection intensity. In *in International conference of genetic algorithm.*

[52] D. Thierens and D. E. Goldberg. Mixing in genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 38–45, 1993.

[53] Ayse Daloglu Vedat Togan. Adaptive approaches in genetic algorithms to catch the global optimisation. In *Seventh international congress advancess in civil engineering, turkey, Oct-2006.*, 2006.

[54] Henry David Venema, Paul H. Calamai, and Paul W. Fieguth. Forest structure optimization using evolutionary programming and landscape ecology metrics. *European Journal of Operational Research*, 164(2):423–439, 2005.

[55] Michele McDonough Venkatraman. Types of task relationships in microsoft project. 2011.

[56] Jing Xiao, Xian-Ting Ao, and Yong Tang. Solving software project scheduling problems with ant colony optimization. *Computers & OR*, 40(1):33–46, 2013.

[57] Zhang and Jeffrey J.P. Tsai. machine learning and software engineering". In *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence*, 2002.

Dinesh Bhagwan Hanchate Birth Place :- Solapur, B.E. Computer from Walchand College of Engineering, Sangli (1995), Lecturer in Gangamai College Of Engineering,Dhule (1995-96), Lecturer in S.S.V.P.S.s B.S.D. College Of Engineering,Dhule In Computer & IT deptt (1996-2005), M.Tech. Computer from Dr. Babasaheb Ambedkar Technological University, Lonere(2002-05), Currently Asst. Prof. Computer Engineering, former H.O.D. (Computer & IT) in Vidya pratishthans College Of Engineering, Baramati , currently doing research (SGGSs Institute of Technology and Engg, Nanded affiliated to SRTMU,Nanded) under the guidance of Dr. Bichkar R.S. ,G.H. Raisonis College Of Engineering and

Management,Wagholi,,Pune.

Dr. Rajankumar S. Bichkar obtained his BE and ME degrees in electronics from the SGGS institute of Engineering and Technology, Nanded, in 1986 and 1990 respectively, and his PhD from IIT Kharagpur in 2000. He served as faculty in the computer engineering and electronics engineering departments in the SGGS Institute of Engineering and Technology from 1986 to 2007, and is presently a professor in the Department of Electronics and Telecommunication Engineering, G H Raisoni College of Engineering and Management, Pune. He has taught several subjects at undergraduate and postgraduate level, including programming in C and C++, data structures computer algorithms, microprocessors, information technology, DBMS and electronic devices and circuits. His research interests include application of genetic algorithms to various search and optimization problems in electronics and computer engineering.