SPS by Combination of Crossover Types and Changeable Mutation SGA

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

Software engineering includes an important 4Ps concept regarding the productivity, processes, people, and project. Efficiently managing skilled people in Software Project Scheduling (SPS), considering various tasks and software project cost is an upheld task. Scheduling and then making Software cost estimation is composite work for project manager (PM) which consists of many cost drivers and their principles related to 4Ps. In this paper, we considered skilled employees as one of the important 4Ps and an important resource to schedule the cost and calculate the cost of the project along with some constraints of tasks. The paper gives a near-optimal estimated cost of project by using different combination of crossover types and dynamic mutation rated Simple Genetic Algorithm (SGA). The paper also considers the aspects of head count, effort and duration calculated by COCOMO-II. These parameters are used to verify the fitness of each chromosome to get estimated cost by SGA closer to the cost estimated by COCOMO-II. The concept of concurrency utilization is included in this paper which satisfies the ultimate aim of project manager or company to get the quality project within minimum time and cost.

General Terms:

Software Engineering Economics, Machine Learning

Keywords:

SGA, COCOMO-II, Software Cost Estimation, Project Scheduling

1. INTRODUCTION

LaTeX The main objective of any software project scheduling is ultimately to reduce the cost of the project. Project planning includes analysis, design, coding and testing phases. Each phase is the collection of umbrella activities in software projects. Every umbrella activity is the collection, flow and gathering of action points [20]. Gathering of action point is one activity. Typical, specific flow of fine and controlled activities output the good solution. This achieves the target of completing every task of umbrella activities. Scheduling of every activity becomes cumbersome for project manager within time limit by considering all the principles of analysis, design and coding phases. This involves all the concepts of 4P, principles of Software Engineering Institute (SEI), process models defined so far, requirement engineering principle, system engineering principle, data models, software tools, technical manual preparation, structural design, procedural and non procedural coding. All these are essential requirement of scheduling in project management. Software engineering provides several process models for project planning. These include iterative, waterfall, RAD, parallel, concurrent etc. Apart from this, a project can be designed in various ways e.g. layered, centralized, distributed and component based design [23]. For each project phase, project plan specifies, resources to fulfil the tasks assigned by project manager within stipulated time and limited budget provided. That can only be possible if PM manages all 4Ps in the project phases with specific and satisfied constrained manner. Several factors play an important role in Software Project Management (SPM) or rather indirectly in scheduling of the projects. These include constraints regarding people working in the project and another actors in the project, skills acquired by the employees, specialized domain of the project and experts, team size and overtime of the employees. It is very difficult to obtain a near optimal solution for scheduling problems. They can be solved by various meta heuristic search techniques. We have used Simple Genetic Algorithm to solve project scheduling problem [3] [8]
The paper is organized as follows

—Section II introduces concept of SPS including software engineering economics and also defines the SPS problem and explains Genetic algorithm.

—Section III discusses SGA-based approach for the solution of the SPS problem.

—Section IV shows the input output tables.

—Section V presents the simulation results.

—Section VI gives the conclusion and future directions.

2. SOFTWARE ENGINEERING ECONOMICS AND PROBLEM DEFINITION

2.1 Software engineering economics

The software engineering depends upon the performance of every P (resource) available which decides the cost and quality of the software product produced [2]. The software engineering becomes valuable due to large and ever increasing costs of the 4Ps (resources) required for software development. The increasing challenges for software professionals in developing and maintaining software require that developed software systems (by computer)

should be most reliable, easy to use, hard to misuse, auditable and keep the people as first person in the development of software instead of computers or software. Another problem or challenge is economic productivity which can be improved by well managed 4Ps available. One of the most important aspects is estimation of the software project cost which has been taken into consideration by various researchers as challenge. They produced many cost estimation models. One of the popular models, well known model, is COCOMO model developed by Bohem [20]. According to the Barry Bohem, software engineering economics must have successful software development processes. He divides the software engineering in two parts namely successful software product and successful software process. Each part has goals of human relations, resource engineering and program engineering.This paper restricts in one of the small parts of resource engineering i.e. planning and estimating.

2.1.1 Significance of Cost Estimation in SPM. Software cost estimation gives very important relation between concepts of economics and software engineering domain. It is plinth and foundation for SPM [2]. Inaccurate cost estimation may lead to following problems.

(1) Software project personnel may have a dilemma of how to schedule a project in unrealistic budget.
(2) Software analyst is not in a position to do the analysis of software-hardware trade-off.
(3) Project manager, most importantly, will not be in position to tell how much duration or time and effort will actually be required for developing or proceeding through each software development phase.

2.1.2 **Effort, COCOMO Models and Estimation**. 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 [2].

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 [9] [19]. 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. Efforts (by COCOMO Model) is assumed and calculated as one of the inputs to the approach described in this paper. 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.

2.2 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.

• Let T be a set of tasks, T={T_i, i=0,....,n-1} where n is the

Table 1. Table showing Notations and their meanings.

Notations	Meaning
MM	man-months
PM	Person-Months
KDSI	Thousands of delivered source instructions
TDEV	Development time
HC	Head Count
GSDH	Schedule derived by our approach using GA
CD $_{T_i}$	COCOMO Calculated duration of task T_i
COD $_{T_i}$	COCOMO Optimistic duration of task T_i, equal to 0.86 × CD
GOD$_{T_i}$	Optimistic duration of task T_i obtained by SGA approach
CPD$_{T_i}$	COCOMO pessimistic duration of task T_i and equal to 1.6 × 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
s^{th}	Schedule number in generation
FC$_s$	Fitnesss of s^{th} Schedule
P_s	Total Penalty of s^{th} schedule
ND$_s$	Normalised duration of s^{th} schedule
SE$_{max}$	Maximum salary of employee among all employees
NHCP$_s$,	Normalised head count Penalty of s^{th} schedule
NTP$_s$,	Normalised total time Penalty of s^{th} schedule
NITP$_{T_i}$	Normalised task incompleteness of T_i task
TCHC$_s$	Total COCOMO head count of s^{th} schedule
GSHD$_s$	Schedule obtained by SGA approach of s^{th} schedule
CPSHD $_s$	COCOMO pessimistic of s^{th} schedule
COSHD $_s$	COCOMO optimistic of s^{th} schedule
TNITP$_s$	Total Normalised Incompleteness task Penalty of s^{th} schedule
20t5e4s	20 Tasks, 5 Employees, total 10 Skills but employees possesses 4-5 skills

Table shows notations and symbols.

Table 2. Relation between effort,duration & source instructions.

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) $^{0.32}$

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 × 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 [11] 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, GSHD 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 [22]. A sample TPG for 10 tasks, 5 employees and 10 skills is shown in Fig 1. There are various types of tasks in a

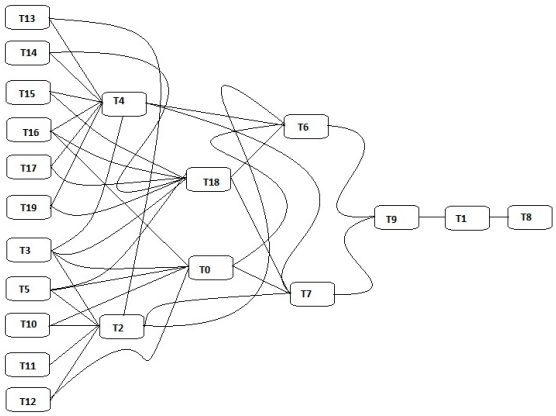


Fig. 1. Task Precedence Graph for 20t5e4s.

project [26]. These are Start to Start (SS), Start to End (SE), End to Start (ES), and End to End (EE). In this paper, SE type tasks are considered. 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. *Using equation1, all the above corresponding durations CPD,COD are calculated.* Pessimistic effort is taken as threshold value and hard constraint in this problem. It is usual and common to put 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.
Here, an indirect constraints handling is used 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 SPSP (Software Project Scheduling Problem) are given and listed bellow.

2.2.1 **Hard constraints**. The tasks in given SPS problem are to be assigned to employees subject to the following hard constraints [4].

(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.

2.2.2 **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 project manager, 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.

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

TPHC is total project head count.
(8) The task duration should not exceed the pessimistic duration (CPD_{T_i}).

$$TDEV_{T_i} = 3 \times EF_{T_i}^{0.328} \quad (9)$$

$$\sum_{j=1}^{e} GSHD_{Ti\,,Ej} \leq CPD_{T_i}. \quad (10)$$

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.

$$\sum_{i=1}^{n}\sum_{j=1}^{e} GSHD_{Ti\,,Ej} > 0.86 \times \sum_{i=1}^{n} COD_{T_i}. \quad (11)$$

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. The 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) [12].

2.3 The Backflow Algorithm

Backflow algorithm is standard simple algorithm to find critical paths time for each vertex of a project digraph. The algorithm is as follows [8] :

(1) Begin at the END vertex of the project digraph and assign it a critical time of zero.
(2) Move backwards to each vertex, that is incident to (having an arrow pointing to) END and assign it a critical time which is the same as the processing time.
(3) For each of the vertices in Step 2, move backwards again to each vertex that is incident to (precedes) it and assign it a critical time ,by adding it's processing time to the LARGEST critical time of the vertices emanating from (having an arrow pointing away from) that vertex.
(4) Continue this process until each vertex of the project digraph has a critical time [3].

2.4 Genetic Algorithm

The Motivation to do the work in GA comes from biological evolution book written by J.H. Holland. The main two objectives of GAs are,

—To apply process of nature systems to a search and optimization problem.
—To make model for maintaining intact natural system mechanisms..

The beauty of genetic algorithm is that it solves multi-constrained, complex problems by its evolutionary technique and yield multiple solutions. SGA has two mechanisms which are dependent on the applied application. These are chromosome representation and evolution operation. A GA [12] is an iterative process having two phases i.e. evaluation and generation.

—Evaluation phase utilizes domain data, constraints and axioms to evaluate the fitness of each individual.
—Generation includes selection and recombination phases.

SGA includes two parts. One is operation and second one is selection. In operation, it includes crossover and mutation with various ways to apply to the problem. Rate of crossover, mutation are the vital and important factors for solving any scheduling problem for satisfying soft and hard constraints [4]. In selection, the problem is how to select these chromosomes. According to Darwin's theory of evolution the best ones survive to create new offspring. There are many methods in selecting the best chromosomes. Examples are roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and some others.
A typical GA pseudo-code is shown in the following sketch.

(1) initialize population;
(2) evaluate population ;
(3) while (convergence not achieved) do {
  (a) scale population fitness;
  (b) select solutions for next population;
  (c) perform crossover and mutation;
  (d) evaluate population;
  }
(4) End.

2.4.1 **Crossover in SGA**. The behavioural of GA depends upon Crossover rate P_c and mutation rate P_m. Crossover rate is the frequency with which crossover is to be processed. If Cs is number of chromosomes in the Population then in every generation, $Cs \times P_c$ chromosomes must do the crossover of selected type. If we have high crossover rate then we must have good selection of individuals rather than more super individual in the next pool. The low crossover rate ultimately lowers the strength of exploration power [9].

2.4.2 **Mutation in SGA**. A unary variation operation, mutation is applied to one genotype and delivers a modified mutant, the child or offspring of it. In general, mutation is supposed to cause a random unbiased change which gives a guarantee of the connectivity search space.

Table 3. A sample example of solution in terms of Schedule representation.

	t_0	t_1	t_2	t_3	t_4	t_5
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

2.4.3 **Selection in SGA** . Generally, it is needed to have two types of selection i.e. minor and major selection. Major selection consists of proportional, ranking, binary tournament selection.There are many minor selection methods such as extinctive, preservative, left and right extinctive, elitist, pure selection methods also [18] [27].

3. PROPOSED SGA APPROACH FOR SPS

3.1 Chromosome Structure

The assignment of tasks to employees is given on the devotion basis. Here,it is the 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

$$Devotion = \frac{g}{G_{max}} \times 100 \quad (12)$$

where g is gene value and G_{max} is max value of gene. String representation of chromosome is used in this 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.

3.2 Solution Representation in SGA approach

We proceed to describe the elements of a solution for the problem. A solution can be represented with a matrix GSHD =(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%.

3.3 Operations for proposed model

3.3.1 Crossover,rate and combination of types. The crossover operator mimics the way in which bisexual reproduction passes along each parents good genes to the next generation [14]. 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 [10].
In proposed approach, following flow of operation of crossover is experimented. The Pseudopod for the same is given bellow.

1. If (getRandom(1.0)$<$ P_c)
Then
a. If Crossover Type = One Point crossover
do One point Crossover(Chrom1, Chrom2);
else
b. If Crossover Type = Two Point crossover
do Two point Crossover(Chrom1, Chrom2);
else
c. If Crossover Type = Multi Point crossover
do Multi point Crossover(Chrom1, Chrom2);
else
d. If (crossover Type = Roulette)
1. Random Number = Random(3);
2. If (Random Number $<$ 1)
a. do One point Crossover(Chrom1, Chrom2);
else
If (Random Number $<$ 2)
b. do Two point Crossover(Chrom1, Chrom2);
else
c. do Multi point Crossover(Chrom1, Chrom2);
End.
End
End.

3.3.2 Mutation. Mutation was done to change in the direction search [15] space as lowest fitness values (continuation) dont provide solutions. We kept on changing the mutation rate [1] with respect to changes in the number of solution's fitness. We generally,increased the P_m if there is increase in the same fitness valued solutions in the current generation [17] [6]. We kept the range of mutation in between 0.01 to 0.05.
Here, in this algorithm presented bellow, g: Number of Generation, 1^{st} Chromosome fitness is maximum due to the ranking method applied in selection.

1. SameFitness= Fitness(1)
2.For chromosomeI = 2 to g
if(SameFitness = Fitness(ChromosomeI)
Same++;
3.P_m = (Same/g)$\times$0.05
4.End.

3.3.3 Stopping Criterion. Though, it is ,generally, expected that with additional generations, solution quality improves, marginal gains decrease. Typically, genetic algorithms terminate after a predetermined number of generations have 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 [25]. Due to the varying pressures of the selection strategies employed, we would expect populations to converge at different times (generation counts) for different strategies. As, we were interested in the temporal (generational) performance of the various selection strategies, we opted to use a maximum generation count as a stopping criterion. Looking at the strategies performance, at different points, in time, enables us to compare convergence properties from a common perspective [16].

3.4 Fitness for proposed approach

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

$$FC_s = \frac{1}{ND_s + P_s} \tag{13}$$

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

$$ND_s = GSHD_s / AD_s \tag{14}$$

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

3.4.1 **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

$$GSHD_s = \sum_{i=1}^{e} \sum_{j=1}^{n} DV_{(E_i,T_j)} \tag{15}$$

where,

$$DV_{(E_i,T_j)} = devotion\, of\, employee\, E_i\, to Task\, T_j\, in\, months. \tag{16}$$

3.4.2 Project (Schedule) Cost. The total schedule cost (SC) or project cost (PC) is obtained by multiplying time required for the project(in months) and salary of all employees per month.

$$SC_s = \sum_{i=1}^{e} \sum_{j=1}^{n} DV_{(E_i,T_j)} \times SE_{E_i} \tag{17}$$

Maximum cost of the project is calculated by,

$$MSC_{max} = \sum_{i=1}^{n} CPSHD_s \times SE_{max} \tag{18}$$

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

3.5 Total Schedule penalty

Total penalty [21] is addition of penalties regarding cost,time,individual task and head count. There are some competing objectives which may give the delay to get the right solution. The example is 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 [4]. 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

$$P_s = NHCP_s + NTP_s + TNITP_s \tag{19}$$

where, $NHCP_s$, NTP_s, $TNITP_s$ are normalised head count, normalised total time and normalised Total task incompleteness penalty of s^{th} schedule respectively.

3.5.1 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. The team size is obtained for current schedule and compared the team size calculated initially by COCOMO-II 1999 model. 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.

$$HCP_{T_i} = \begin{cases} 0 & if Condition1 \\ |CHC_{T_i} \pm 1 - GHC_{T_i}| & Otherwise \end{cases} \tag{20}$$

$$Condition1 \equiv CHC_{T_i} \pm 1 = GHC_{T_i} \tag{21}$$

$$NHCP_s = \frac{\sum_{i=1}^{n} HCP_{T_i}}{TCHC_s} \tag{22}$$

where $TCHC_s$ is total COCOMO head count of s^{th} schedule.

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

$$TP_s = \begin{cases} rate * (CPSHD_s - GSHD_s) & if\ Condition1 \\ 1 \times 10^9 & if\ Condition2 \end{cases} \tag{23}$$

$$Condition1 \equiv COSHD_s <= GSHD_s <= CPSHD_s \tag{24}$$

$$Condition2 \equiv GSHD_s > CPSHD_s \tag{25}$$

Where,rate is penalty per unit time (rate is kept as 1 penalty/months), GSHD is schedule obtained by SGA 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.**

$$NTP_s = \frac{TP_s}{CPSHD_s} \tag{26}$$

NTP_s is Normalised time penalty.

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

$$\sum_{i=1}^{n} NITP_{T_i} = \sum_{i=1}^{n} (GOD_{T_i} - COD_{T_i}) / CPD_{T_i}. \tag{27}$$

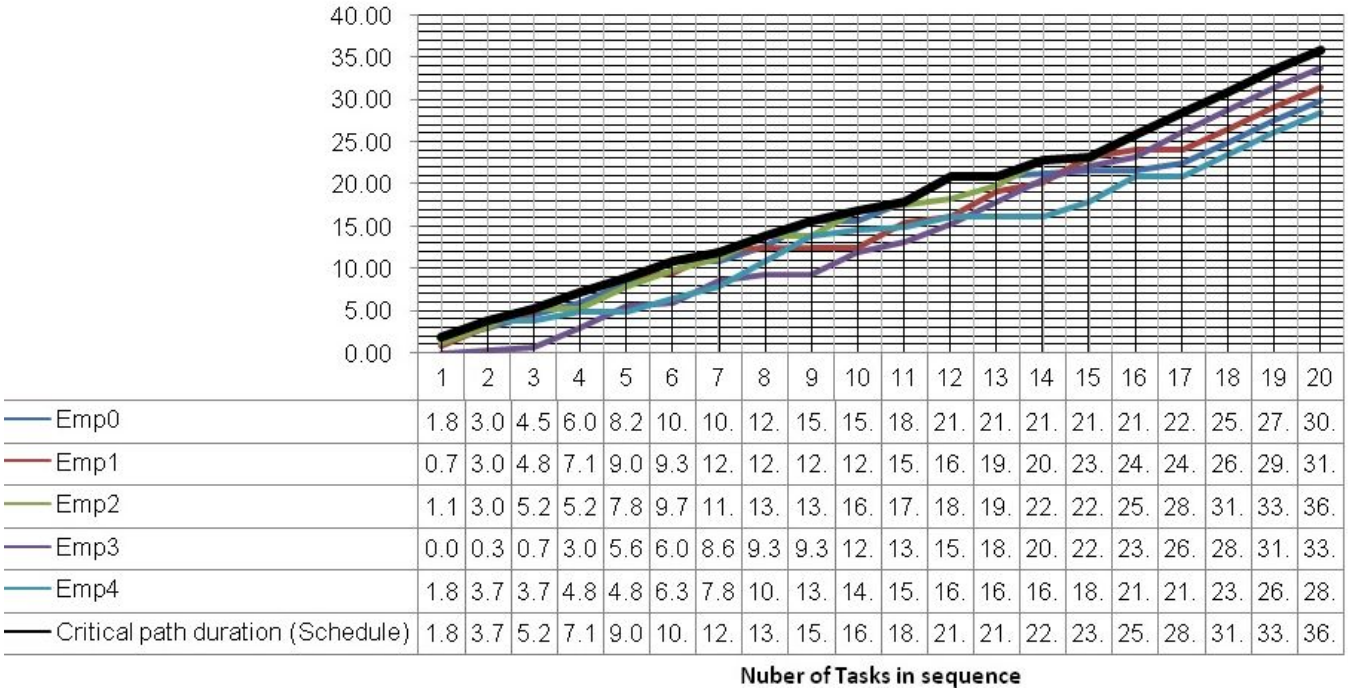
where, CD_{T_i} is COCOMO Calculated duration, COD_{T_i} is COCOMO Optimistic duration & equal to 0.86 $\times$ CD, GOD_{T_i} is Optimistic duration obtained by SGA approach, CPD_{T_i} is COCOMO pessimistic duration of task T_i and equal to 1.6 $\times$ CD (for task T_i).

$$TNITP_s = \sum\nolimits^{n}{}_{i=1} NITP_{Ti} \tag{28}$$

The incompleteness is normalised by above equation.

4. INPUT AND OUTPUT

The input as configuration files [24] is taken, which has already been given by Jing,Tang and Ting. Nearly, 30 configuration files have been given. Each file contains mathematical notation written as instructional statements as shown in Table-III. These files are read for collecting employees properties, task properties and TPGs. INPUT tables shown in the papers are having various parameters which is used as inputs to propsed model and OUTPUT tables indicates various output parameters useful for obtaining the schedule. The input configuration file is shown in table-IV. Input parameters are taken from Table-V,VI. Output tables are Table-VII,VIII and Table-IX.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Emp0	1.8	3.0	4.5	6.0	8.2	10.	10.	12.	15.	15.	18.	21.	21.	21.	21.	21.	22.	25.	27.	30.
Emp1	0.7	3.0	4.8	7.1	9.0	9.3	12.	12.	12.	12.	15.	16.	19.	20.	23.	24.	24.	26.	29.	31.
Emp2	1.1	3.0	5.2	5.2	7.8	9.7	11.	13.	13.	16.	17.	18.	19.	22.	22.	25.	28.	31.	33.	36.
Emp3	0.0	0.3	0.7	3.0	5.6	6.0	8.6	9.3	9.3	12.	13.	15.	18.	20.	22.	23.	26.	28.	31.	33.
Emp4	1.8	3.7	3.7	4.8	4.8	6.3	7.8	10.	13.	14.	15.	16.	16.	16.	18.	21.	21.	23.	26.	28.
Critical path duration (Schedule)	1.8	3.7	5.2	7.1	9.0	10.	12.	13.	15.	16.	18.	21.	21.	22.	23.	25.	28.	31.	33.	36.

Fig. 6. Graph shows assignment of employee, duration for each employee to tasks, critical path duration, total duration for 20t5e4s example

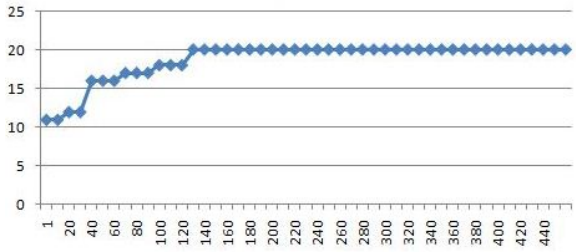


Fig. 2. Generation Vs Task for 20t5e4s example

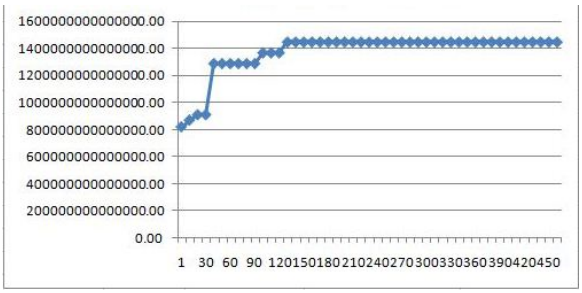


Fig. 4. Generation Vs Fitness for 20t5e4s example

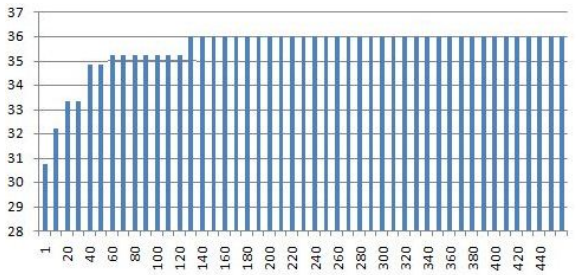


Fig. 3. Generation Vs Time Completed for 20t5e4s example

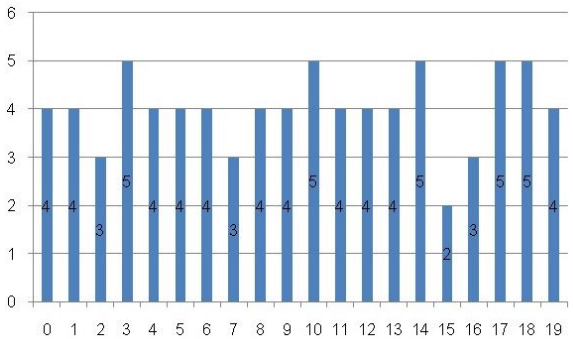


Fig. 5. Task Vs Number of Employees for 20t5e4s example

5. RESULT AND DISCUSSION

The proposed model is implemented in Java with GA Java Classes in netbean, the window based IDE environment. Nearly, all configuration files were taken and conducted to see the accuracy and complexity of model and did compare with Jing, Tang and Ting's results [24].

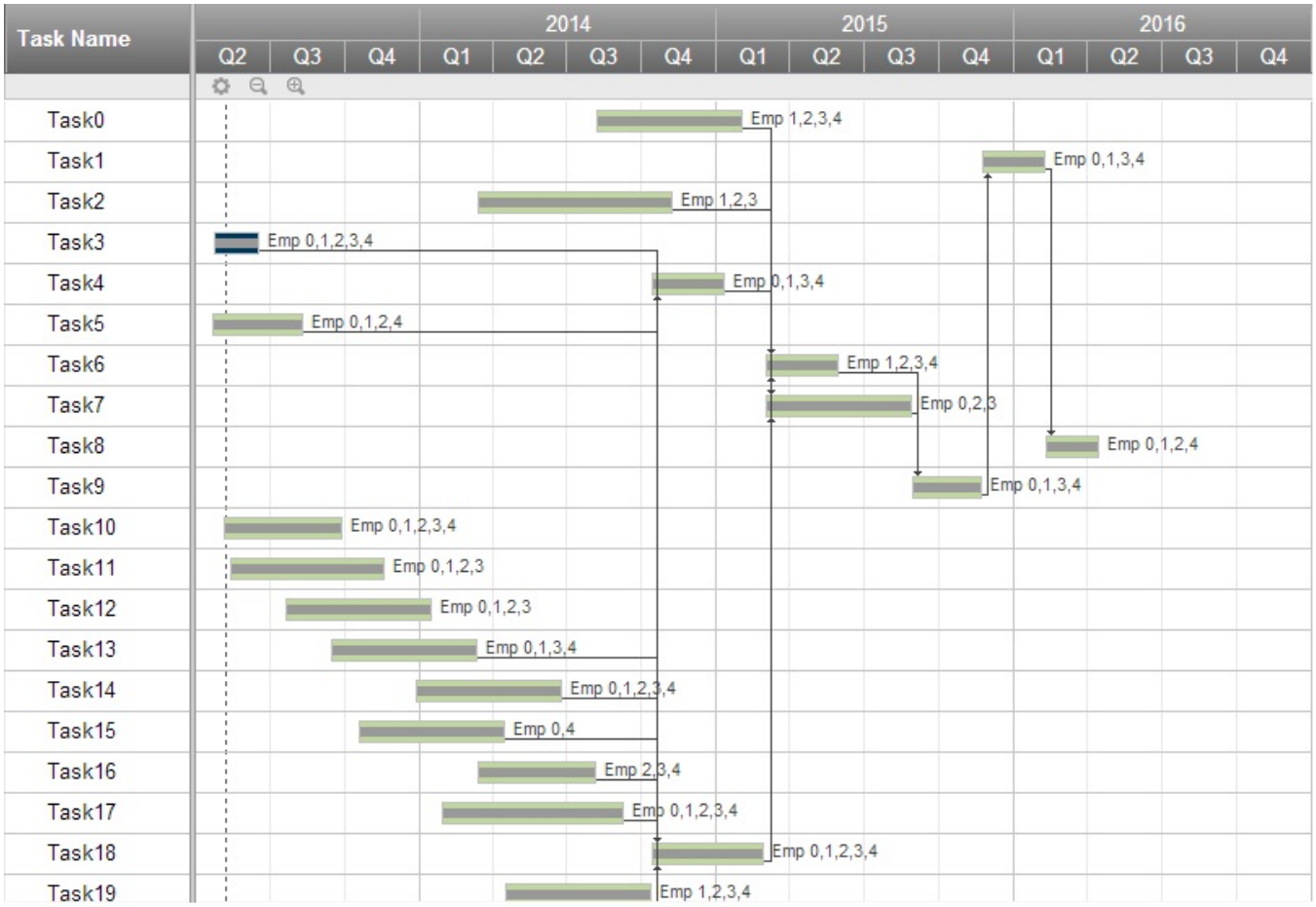


Fig. 7. Gantt Chart for Task Precedence Graph for 20t5e4s example

5.1 Parameters of GA

It is observed from some researchers as well by us also, that [13] P_m, generations and pool size do affect on time required by GA. Right from group of 5 employees to 20 employees with varying in the skill numbers and tasks, we experimented on all by making changes in the above parameters. It is observed that more population size and generations give best results for small size than big size problems. An emphasize is given on number of more generations as better performance can be achieved by making generation numbers large in big size problem too. There is an effect of mutation probability on the behaviour of model. That's P_m is kept on changing as per the solutions obtained from the generation to generation. Another aspect of constraints had been observed and studied i.e. with respect to number of soft and hard types constraints. We felt that increase in the tight constraint make the search space bounded and making it hard to groom the population. Very tight restriction needs more and more generation to satisfy and achieve the better solution. Larger space can be obtained due to less constraints and got better chromosomes as well. However, 30 to 45 minutes are required to get the better solutions from small size to large size input configured parameters, set by Ting and Tang.

5.2 Experimental Result and Discussion

This approach is used the techniques such as static and constant penalty method. The both penalty and reward are taken for the components 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 all hard and but, nearly all soft constraints.

In the problem, the effort, duration and head count defined by Bohem is considered for making correction in the fitness value of individual schedule and implementation as well. The experimental results of all the works shows that crossover rate 0.9 gives better solution than 0.6. The mutation rate 0.01 gives good results but requires much and more time. If we increase the mutation rate, the search space goes in the direction of randomization. The types of crossover also affects on the solution as if, one point crossover is taken for operation then it gives less probability of combinations of genes. If multi-point crossover [27] is taken, the two side of chromosome attach each others end and do the crossover at multi-point. Ranking selection method is used as major selection method. The combination of all these minor selections are studied. In preliminary run, the right extinctive selection is used for 2 to 5 individuals (having less fitness and having zero production rate) for

Table 4. INPUT:Configuration file showing mathematical notations for 10 Tasks, 5 Employees and 4 Skills.

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
employee.1.skill.3=1	employee.0.skill.1=3
employee.1.skill.2=8	task.2.skill.number=2
employee.2.salary=9935	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	

Table 5. INPUT: Employee properties for 20t5e4s sample input example.

Emp Id	Salary/month	loading limit	Employee skills
0	9793	1	0,1,5,7,9
1	9545	1	0,1,6,8,9
2	10131	1	1,2,4,8
3	10252	1	0,5,6,9
4	10944	1	0,3,7,8

some generation to discard the some but little number of chromosomes which dont tend to give super-individuals. Elitist selection scheme is used which enforces to go through selection along with their parent by making some individual duplicates to be inserted in next generation. The recombination of all different directed quality chromosomes are tried as the better solutions are obtained through the combination of all the levels of minor selections of individuals. Though, we concentrate on hybridization of selection process but biologists takes mutation as main source of evolution.
In preliminary run, the pool of chromosome is set 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, some good solution is obtained in chromosome pool to propagate to the next generation. Figure 3 to

Table 6. INPUT : Task Properties for 20t5e4s example but employees possesses 4-5 skills.

Task ID	Effort (PM)	CD (Months)	CPD (Months)	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 7. OUTPUT: Tasks durations as division of tasks in Months for employees and total duration for every tasks by SGA approach for a 20t5e4s example .

	Emp0	1	2	3	4	GOD
Task0	0.38	1.13	3.00	2.63	0.00	7.13
Task1	2.63	1.13	0.00	1.13	0.75	5.63
Task2	0.00	3.00	1.50	2.63	0.00	7.13
Task3	1.13	2.25	1.88	0.38	1.88	7.50
Task4	0.38	3.00	0.00	1.50	1.88	6.75
Task5	1.88	0.75	1.13	0.00	1.88	5.63
Task6	0.00	0.75	3.00	1.13	3.00	7.88
Task7	0.75	0.00	2.63	3.00	0.00	6.38
Task8	2.25	2.25	1.13	0.00	1.88	7.50
Task9	2.63	1.88	0.00	1.88	1.50	7.88
Task10	2.63	0.38	1.88	0.38	1.50	6.75
Task11	1.50	1.88	2.25	0.38	0.00	6.00
Task12	2.25	1.88	2.63	2.63	0.00	9.38
Task13	1.50	2.25	0.00	2.25	1.13	7.13
Task14	1.88	0.38	2.63	0.75	3.00	8.63
Task15	3.00	0.00	0.00	0.00	3.00	6.00
Task16	0.00	0.00	3.00	2.63	0.75	6.38
Task17	2.25	3.00	0.75	1.13	0.38	7.50
Task18	3.00	0.75	0.75	2.25	1.13	7.88
Task19	0.00	2.63	1.50	2.63	1.50	8.25

7 gives different graphs. These graphs gives the behaviour of SGA with parallelism in scheduling. 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 [7]. If all tasks are completed first within group of sequenced parallel tasks, in turn, one 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 dis-

Table 8. This table shows that all tasks are completed as COD $\leq$ GOD $\leq$ CPD for 20t5e4s.

	CD	COD	GOD	CPD
Task0	6.59	5.66	7.13	10.54
Task1	4.30	3.70	5.63	6.88
Task2	6.78	5.83	7.13	10.84
Task3	5.93	5.10	7.50	9.49
Task4	6.78	5.83	6.75	10.84
Task5	5.68	4.88	5.63	9.09
Task6	7.29	6.27	7.88	11.67
Task7	8.14	7.00	6.38	13.03
Task8	6.59	5.66	7.50	10.54
Task9	7.74	6.66	7.88	12.39
Task10	6.38	5.49	6.75	10.22
Task11	5.93	5.10	6.00	9.49
Task12	7.60	6.53	9.38	12.16
Task13	7.29	6.27	7.13	11.67
Task14	7.45	6.41	8.63	11.92
Task15	5.68	4.88	6.00	9.09
Task16	6.38	5.49	6.38	10.22
Task17	6.38	5.49	7.50	10.22
Task18	6.59	5.66	7.88	10.54
Task19	7.29	6.27	8.25	11.67

Table 9. Employee working times for 20t5e4s.

Employee	Time	No. of Task Per Employee
0	30.30	16
1	23.65	17
2	29.92	15
3	25.05	17
4	25.72	15

Table 10. Comparison Table of project duration between SGA approach and Jing, Ting and Tang's results in months duration for 20t5e4s example.

Group	OUR Result (GCD in months)	Jing, Ting and Tang's results in months
10t5e2s	22.25	24.26
10t10e4s	12.375	14.21
10t10e6s	13.875	13.25
10t15e4s	9.1	8.23
10t15e6s	9	8.03
10t15e2s	8.87	8.14
20t5e6s	56.25	58(ACS)
20t5e2s	56.575	58(ACS)
20t10e4s	35.625	37.89
20t10e6s	37.50	37.99

Table shows various configuration inputs. Total skills are 10. Out of 10 skills 2-3, 4-5, 6-7 skills are used. Here in this table, 2s, 4s and 6s means 2-3, 4-5 and 6-7 skills respectively.

tribution 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 11. Xover Computation (after which fitness remains nearly same) for 20t5e4s example.

Xover Methodology	Avg (SGA) Computation
One point	210
Two pont	160
Uniform	240
Selective	290

5.3 Computational SGA results

The results for the sample example case for the given 20 task, 05 employee and 10 skills are shown in all the table-VII to table-XI. 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 value. Though, the fitness gives constant value but it gives some different solutions by considering all the composite components of the schedule. Combination of different crossover type and rate, mutation rate are put according to number of same fitness values of chromosomes. The generation is set to 500 for this specific problem of 20 Tasks, 5 Employees, 10 Skills.
The following are the outputs obtained as result of implementation.

—schedule as 2D chromosome in terms of Employee Task assignment.

—conversion of these matrix in terms of months assigned as per an employee devotion for task (assigning of the each employee to the task as per the TPG and sequence).

—Time required for tasks, employees working time.

—Getting the distribution of employee over the number of tasks.

—Team size to complete an each task.

—Parallel adjustment of each employee(Every task is assigned to skilled employees).

—Time required to complete the project i.e. schedule.

—Cost of the schedule

—Fitness of the schedule.

—Gantt chart of proposed SGA example.

5.4 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.

5.5 Comparative discussion

Here, The results of this model with Ting,Tang's [24] results are compared. Somewhere, it has been found that the result of ACO (Ant colony optimization) are better and some time proposed approach's results are. This observation can also be seen in the results of Ting,Tang. ACO is better than SGA for some of the smaller problems. The proposed approach SGA put behind the GA result of Ting and Tang, nearly in 50% of all results. Proposed SGA 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 the fitness of individual. The proposed approach results produced may give hope of making project manager feel satisfy about the schedule.

6. CONCLUSION AND FUTURE WORK

The objective was to solve the scheduling in minimum time and cost with minimum penalty which is achieved comparatively good. Software engineering also gives some future scope to this problem using some data model approach to reduce the effort in software project using parallel genetic approach by running parallel projects is huge multi-project for the PM. Future work gives opportunity to do scheduling with parallelism between the module to module or project to project. Multi-project scheduling problem is challenging issue using various types of genetic algorithms [5].

7. REFERENCES

[1] Charles F Baer. Does mutation rate depend on itself? In *PLoS Biol*, Feb 2008.

[2] Barry Bohem. *Software Engineering economics*. Prentice Hall PTR, New Jersey, 1981.

[3] Dr. Larry Bowen. Scheduling algorithms. 2001.

[4] Mark Christensen Carl K. Chang. A net practice for software project management. In *IEEE software*, Nov-Dec 1999.

[5] Avraham Shtub Cohen, Avishai Mandelbaum. Multi-project scheduling and control: A process-based comparative study. In *Project Management Journal*, JUNE.

[6] Charlesworth B Charlesworth D Crow JF Drake, JW. Rates of spontaneous mutation. In *Genetics*, page Volume 5, 1998.

[7] E.W.Davis and G.E. Heidorn. An algorithm for optimal project scheduling under multiple resource constraints. In *Management Science*.

[8] Hilary Freeman. Importance of tasks. 2001.

[9] David E. Goldberg. *Genetic Algorithm in Search, Optimization and Machine Learning*. Addison-Wesley, New Jersey, 1989.

[10] David E. Goldberg and Kalyanmoy Deb. *A comparative analysis of selection schemes used in GAs.* IOP Publishing Ltd and Oxford University Press, UK, 1997.

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

[12] Sam Hsiung and James Matthews. An introduction to genetic algorithms.

[13] Jalote. *An Integrated Approach to Software Engineering.* Narosa publishing House, UK, 2005.

[14] J.J.Grefenstette. Optimization of control parameters for genetic algorithms. In *IEEE Trns. System, Man and Cybernetics*, Jan 1996.

[15] PD Keightley. Rates and fitness consequeces of new mutations in humans. In *Genetics*, pages 295–304, Feb 2012.

[16] Pukkala M. Kurttila. Examining the performance of six heuristic optimisation techniques in different forest planning problems. In *Trends in Ecology and Evolution*.

[17] Park Mark Anclif. Mutation rate threshold under changing environments with sharp peak fitness function. In *Journal of the Korean Physical Society*, page Volume 5, 1993.

[18] Tom Mitchell. *Machine Learning*. McGraw-Hill, New Jersey, 1997.

[19] Pinedo. *Scheduling:Theory, Algorithms, and Systems*. Kluwer Academic Publishers, UK, 2001.

[20] Roger S Pressman. *Software Engineering: A practitioners Approach*. McGraw-Hill Inc., New york, 2001.

[21] Jeff D. Hamann P.Thompson and John Sessions. Selection and penalty strategies for genetic algorithms designed to solve spatial forest planning problems. In *International Journal of Forestry Research*, page 14, 2009.

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

[23] Ian Sommerville. *Software Engineering*. Addison-Wesley, New york, 2011.

[24] Ting and Tang. Solving software project scheduling problems with ant colony optimisation. In *Computer and Operation Research*, 2013.

[25] P. H. Calamai Venema and P. Fieguth. Forest structure optimization using evolutionary programming and landscape ecology metrics. In *European Journal of Operational Research*.

[26] Michele McDonough Venkatraman. Types of task relationships in microsoft project. In *Venkatraman,Michele McDonough, Types of Task Relationships in Microsoft Project*, pages Lesson–5, August 2011.

[27] Liao Ying-Hong and Chuen-Tsai Sun. An educational genetic algorithms learning tool. 2001.

8. BIOGRAPHY

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.