A Hybrid Technique for Software Project Scheduling and Human Resource Allocation

A. Avinash, Dr. K. Ramani
Department of Information Technology
Sree Vidyanikethan Engineering College, Tirupati

***Abstract* – Project planning plays a crucial role in software project implementation and to minimize the cost. The existing models suffer from a large search space and lack of flexibility in human resource allocation to simplify the model. In the proposed model, the tasks identification and scheduling is implemented using Event-Based Scheduler (EBS) and efficiency of employees and allocation of tasks is done through Ant Colony Optimization (ACO). Through EBS, beginning and ending of the tasks can be considered as events and it enables optimized resource utilization. The Task Precedence Graph obtained through ACO is used to find shortest path which reflects most suitable employee for the given task. Combining EBS and ACO techniques together will minimize the search space for employee allocation and hence the cost of the given project. The experimental results show that the proposed method is efficient and effective.**

***Keywords* - Project scheduling, Resource allocation, Task Precedence Graph, Ant Colony Optimization and Software project planning**

I. INTRODUCTION

The software industries have to make efficient project plans to reduce the cost of software construction. To plan a software project, the project manager needs to estimate the project workload, cost and decide the project scheduling and human resource allocation. Assigning of employees to the best fitted tasks is a challenging job for project managers. Techniques like Program Evaluation and Review Technique (PERT) [15] and Critical Path Method [16] (CPM) also failed to construct the resource allocation and scheduling models like Resource Constrained Project Scheduling Problem (RCPSP) [17]. Therefore, the task scheduling and human resource allocation has treated as two different activities. Existing research shows that task pre-emption can reduce the time and cost of the software project. But, it reduces the flexibility of human resource allocation in project planning.

In software projects, it is common that a programmer is involved in multiple module development tasks simultaneously. Therefore, based on the urgency of other tasks, the task pre-emption has to be designed properly and human resources have to be made more efficient way.

Many software engineering activities like module clustering, cost estimation, design, testing, and software release planning has stated as events and search-based approaches can be used for software project planning.

II. RELATED WORK

C. K. Chang et al. [1] proposed Genetic Algorithms (GAs) for project scheduling, in which weights are assigned to edges of Task Precedence Graph. Then it selects two parent solutions and swaps over feature from each to solve task scheduling problem in software project management. It calculates the complexity of each solution and selects the weighted objective function to get single solution. This helps the engineer to work with bug free and reliably. But continuity of allocation is not supported.

Carl K. Chang et al. [2], proposed a Software Project Management Net (SPMNet) to model software development projects. It helps to allocate the resources and can do project scheduling automatically based on the skills and experience. It creates plan to develop and estimates the schedules and budgets. The automated solutions are generated by using the Task Precedence Graph (TPG). The Genetic Algorithms help to have graph of human resources by retrieving details from the database. But the Task Precedence Graph included is a cyclic one which will be executed infinitely unless the manager specifies a certain number of iterations.

Thomas Stutzle et al. [3], proposed Max-Min Ant System (MMAS) for travelling salesperson problem. To exploit the best solutions during the search and direct the ants to high quality solutions. In this, the Max-Min ant system will limit the pheromone trail to prevent premature convergence. The MMAS failed to manage the best managing solutions of pheromone trails. This technique is included in the proposed approach by rectifying the previous problem to manage the best managing solutions.

Ahilton Barreto et al. [12], proposed an optimized based approach to support resource allocation of a software project which includes project activities, human resources available. In this approach, some of the professionals are taken into consideration with their available schedules, Maximum cost and Minimum cost for their working hours. The goal is to allocate the cheapest team for the project. Each team is assigned for same project to find the efficient and cheap team. But in this approach, the study states that there is no difference between the time to perform the activity and to choose the team.

An effective and efficient automated acyclic Task Precedence Graph is required for identifying Task priority and allocation of suitable employees to tasks based on their skills and experience with least cost. In the proposed technique Event Based

Scheduling and Ant Colony Optimization techniques are used to provide software project Scheduling and human resource allocation.

III. EVENT BASED SCHEDULER

In software development, to manage employees one has to maintain the database with the records of employee's salaries, skills, and working constraints. The main problem is to assign the employees to suitable tasks so that tasks can be done efficiently and effectively. Wei-Neng Chen et al. [4], explained Event Based Scheduler (EBS) for Employee allocation table preparation based on events: Start of the Task, End of the Task, and Employee contribution of that task. According to this, suppose 'n' employees are involved in the project, for i^{th} employee (i=1, 2, 3,..., n) the following attributes are considered.

bs_i– The basic salary for employee per time period.
hs_i– The salary for employee's per-hour normal work.
ohs_i– The salary for the employee's per- hour over time work.
nh – Legal normal working hours per month.
max_i– Maximum possible working hours per month of the employee for the project.
$[join_i, leave_i]$ – The time window when the employee is available for the project.
$\{s_i^1, s_i^2, ..., s_i^{\phi}\}$ – The skill list for the employee.
$hours_i^t$– Employee devoted working hours for t^{th} month.

The employees will be categorized into two types: the regular employee and the temporary employee for particular time period in the project. Regular employee salary will be calculated based on regular working hours and extra amount will be paid for extra hours spent by him beyond the regular working hours. Temporary employee's salary will be calculated on hourly basis. The salary $salary_i^t$ for i^{th} employee at t^{th} month will be calculated by

$$salary_i^t = \begin{cases} bs_i + hours_i^t . hs_i, & hours_i^t \le nh, \\ bs_i + nh. hs_i + (hours_i^t - nh). ohs_i, & nh < hours_i^t \le maxh_i. \end{cases} \quad (1)$$

In a software project, tasks can be any activity involved in software construction. Here the TPG (Task Precedence Graph) is used for selection of task. A task can start only when the direct predecessors of the tasks have been finished. For a task t_j (j=1,2,...,n) the following attributes are considered[4].

pm_j–The estimated work effort of task in person months.
SK_j – The set of skills required by the task.
$maxhead_j$– The maximum headcount for the task.
$deadline_j$ and $penalty_j$– The deadline and penalty of the task.

Based on the above definitions, for suppose wh_{ij}^t is the number of working hours of i^{th} employee for t_j at t^{th} month, the achievement A_j^t of each employee for t_j at time t is evaluated by following steps:

I. The proficiency $prof_{ij}$ of i^{th} employee for t_j is evaluated by

$$prof_{ij} = \prod_{id \epsilon SK_j} \frac{s_i^{id}}{5} \quad (2)$$

II. The total fitness F_j^t of employee for t_j on t^{th} month is evaluated by

$$F_j^t = \frac{\sum_{i=1}^{m} prof_{ij}.wh_{ij}^t}{\sum_{i=1}^{m} wh_{ij}^t} \quad (3)$$

III. Convert the Fitness to cost driver value

$$V = 8 - round\ (F_j^t.7+0.5) \quad (4)$$

IV. The achievement A_j^t for on t^{th} month is evaluated by

$$A_j^t = \frac{\sum_{i=1}^{m} wh_{ij}^t}{V} \quad (5)$$

Event Based scheduler is used for creating events. In EBS the employee planned working hours are calculated and represented as employee allocation matrix [5]. The EBS makes new assignments at events. The time t is considered as an event if t satisfies any one of the following conditions:

1) Starting of the project
2) Any employee joins or leaves the project
3) Any task just finished in the previous time period and the next resources become released and available at 't'.

The EBS is used to adjust the plan in the form of working hours by two rules. First, if there is any resource conflict between two tasks, the task that appears before the task having higher priority to use resource. Second, the new workload assignments are made only when events occur.

Then each event is scheduled with the workloads in chronological order with respect to the growth of time t. If t is an event, then the scheduler will reassign the actual workloads according to the urgency of task recorded in task list. After allocation of actual workloads at t, the achievement A_j^t of t_j at t is calculated. If any of the task in list is finished then the time will be incremented with 1 i.e., t+1 is set as an event.

First, let the regular employee will dedicate his normal working hours to the project. So the costs for hiring extra employee will be saved. The employee normal working hours is planned to dedicate only for time t, but not all his working hours. By this, the task t_j can be found from task list that satisfies $wh_{ij}^t > 0$ and can update the working hours to

$$wh_{ij}^t = wh_{ij}^t + (maxh_i - hours_i^t), \quad if\ maxh_i\ nh,$$
$$wh_{ij}^t + (nh - hours_i^t), \quad if\ maxh_i \geq nh. \qquad (6)$$

This is one way to dedicate all his normal working hours to the project.

Second, if task t_j is finished at time *t*, there may be another workload released. To minimize the cost, it is necessary to reduce the workload assignments. Here are some steps to follow for cost minimization:

1. Arranging the employees in ascending order of their proficiency scores in particular task t_j.
2. Select the employee from the list and denote the employee as *u*.
3. Set the working hours as,
   $wh_{ij}^t = wh_{ij}^t - nh.unitpercent\%$
   And test the task t_j will be complete at time *t* until $wh_{ij}^t = 0$ or t_j cannot complete.
4. If $wh_{ij}^t = 0$, then select the next employee from the list and repeat the *steps* 1 & 3. If task $t_j$cannot complete at time *t*, then reset the working hours of that employee $wh_{ij}^t = wh_{ij}^t + nh.unitpercent\%$.

In this way, the tasks are assigned to each employee based on their proficiencies.

IV. ANT COLONY OPTIMIZATION

The Ant Colony Optimization is an approach based on the behavior of ants developed by Dorigo et al. [6], [7]. When ants are in search of food, they release the chemical called Pheromone to guide the other ants to find the food. In the same way here the pheromone values are uploaded for past experience of each employee for a particular task. In ACO (Ant Colony Optimization), the pheromone will act as a record of past search experience of each employee. From the past experience the components of best solutions gathered by previous ants will usually attract more ants to select those iterations in the future. However, the construction of the task list is a difficult task, the pheromone values may used to get higher probabilities to select the accurate components in solution construction procedure.

In ACO the local search procedure is introduced in the proposed approach. When the various ACO variants have been developed [8], two best performing ACO variants are included, namely ACS (Ant Colony System) [7] and MMAS (Max-Min Ant System) [3]. Here ACS applies a pseudorandom proportional selection rule for selecting the components with maximum pheromone and heuristic values. During pheromone management procedure, the two pheromone updating rules, namely, the local updating rule for reducing the pheromone on components selected by ants to maximize the search diversity of the algorithm and the global updating rule for selecting the components related to best-so-far solution.

In ACS, there are two steps to overcome the problems, including tasks list and employee allocation matrix.

1. Construction of Task List
2. Construction of Employee Allocation Matrix

Construction of Task List

A task list is an order of tasks that satisfies the precedence constraints defined by the TPG. To build a task list, first one has to determine the order of tasks. Since one task has to be allocated to many employees, one employee can work with many tasks simultaneously, and skill proficiency of each employee is considered, it is difficult to allocate the related tasks for related employee. So, here the absolute position model with the summation rule is used in the proposed approach.

The MINSLK (Minimum Slack) heuristic [9], [10] is adopted for the construction of task list. Actually a task with smaller MINSLK will be denoted as a preferable task in the list. The MINSLK for task $t_j$can be estimated as follows:

1. Estimate the shortest possible makespan of each task.
2. Based on the shortest possible makespan the earliest start time and the latest start time of each task will be evaluated and the MINSLK is calculated by the difference between latest start time and earliest start time of each task. Then the heuristic for task t_j is evaluated by

$$\eta_t(j) = 1/MINSLK_j \qquad (7)$$

To build a task list, each ant maintains an $eligibleSet$of tasks that satisfies the precedence graph. The construction of task list includes the following steps:

1. Select the task that can be implemented at the beginning of the project and insert in $eligibleSet$.
2. For *k*=1 to *n*, process the steps 3 & 4 repeatedly.

3. Select the task from the $eligibleSet$ and arrange the task in the k^{th} position of task list. Using selection rule, random number q distributed in [0, 1] is generated randomly and compared with a parameter q_0. If $q<q_t$, then task t_j from the $eligibleSet$ which is having the largest value is chosen to arrange the task in k^{th} position.
4. The probability of selecting the task t_j to k^{th} position is given by

$$\Pr(j,k) = \begin{cases} \frac{\sum_{l=1}^{k} T_t(j,l).\eta_t(j)}{\sum_{t_u \epsilon eligibleSet} \sum_{l=1}^{k} T_t(u,l).\eta_t(u)}, & if\ t_j \epsilon eligibleSet \end{cases} \quad (8)$$

5. Remove the selected task from $eligibleSet$ and update the $eligibleSet$ by adding new feasible tasks that satisfy the precedence constraint into $eligibleSet$.
   After completion of Steps 3 & 4 , repeat 'n' times to build the complete task list.

Construction of Employee Allocation Matrix

The employee allocation matrix consists of planned working hours of each employee to tasks [1], [11]. Here, $nh.unitpercent\%$ is used as unit of working hours.

If $unitpercent\%$=25, the search domain pwh_{ij}becomes {0,25%nh,50%nh,..., $maxh_i$}.Here two kinds of pheromones are updated for the construction of employee allocation matrix. First, for choosing the i^{th} employee to work for task $t_j$which is denoted as $\tau_{e1}(i,j)$. Second, for assigning k (k=25%nh,50%nh,..., $maxh_i$) of i^{th} employee's working hours to $t_j$is denoted as $\tau_{e2}(i,j,k)$.

The heuristic of choosing the i^{th} employee to work for the task t_j is evaluated by

$$\eta_e(i,j) = prof_{ij}/hs_i \quad (9)$$

Here the employee will be chosen based on his high proficiency score and low salary. By the reference of heuristic value, the following steps are used to build employee allocation matrix.

1. All the values in employee allocation matrix have to set to '0'.
2. Assign the workloads for each task t_j.
3. For all employees, evaluate the value of
$$\tau_{e1}(i,j).\eta_e(i,j)^{\beta} \quad (10)$$
4. Select the employee u, which has not chosen for any task using a pseudorandom proportional rule in ACS [7]. The probability of selecting the employee u for task $t_j$is,
$$\Pr(i,j) = \begin{cases} \frac{\sum_{l=1}^{k} \tau_t(j,l).\eta_t(j)}{\sum_{t_u \epsilon eligibleSet} \sum_{l=1}^{k} \tau_t(u,l).\eta_{t(u)}}, & if\ t_j \epsilon eligibleSet, \\ 0, & otherwise. \end{cases} \quad (11)$$
   When an employee is selected based on pheromone values, the employee will be deleted from the Eligible Set.
5. For selected employee the working hours are allocated using pseudorandom proportional rule.
6. When all the employees in the $eligibleSet$ are selected and allocated for certain tasks, it can be said that employees to work for tasks reached maximum headcount.

V. IMPLEMENTATION

In the proposed approach, for experimental purpose 10 Employees and 10 Tasks are considered for project scheduling. The proposed hybrid technique is implemented in Java using Eclipse IDE.

The tasks identified for the project development are as follows:

Table1. Task List

Task ID	Task Name	Task Description
T1	Requirements gathering	Collecting requirements from user
T2	Planning and Analysis	Creating a Project scope and Timelines
T3	Technical Specifications	Develop technical specification for each use case
T4	Project Architecture	Creating a high-level project architecture
T5	Project plan	Creating project plan
T6	Development	Relive's Low-Level Documents (LLD) and writing code
T7	Test Plan	Creating test plan (Timelines, Estimation)
T8	Test Case	Creating test case documents like LLD's
T9	Maintenance	Reviewing the documents and delivering the project to user
T10	Feedback	Collecting the feedback from user

The tasks which are not having the predecessors will be selected and initialized to '0'. The tasks that contain predecessors will be selected and put into a task list (Eligible Set) which is arranged according to the priority of each task. Once the task list is built the complete eligible set is checked to find whether all the tasks are selected or not. Once all tasks are inserted into $eligibleSet$, the employee should be allocated to each task. To select the highest priority task which has high pheromone and heuristic values the pseudorandom proportional rule is used.

Each employee will have some set of skills as follows:

Table2. Skill Set

Skill ID	Skill Description
S1	Able to organize tasks in the right order to hit the right outcome at the right time(Scheduling)
S2	Able to find the right people and negotiate with their managers to ensure that they are available at the right time to work on their project tasks.
S3	Able to resolve the needs and roadblocks
S4	Able to test the people inside the team
S5	Able to motivate, coach, inspire others and manage a team with all interpersonal skills
S6	Able to handle the changes in the project
S7	Able to understand the processes, tool requirements and data requirements for the technologies
S8	Must assist in developing procedures to install the tools
S9	Should be good at splitting work into pieces
S10	Able to manage time(Prioritize, plan, prepare and prevent problems)
S11	Able to manage conflict
S12	Able to provide technical support for technologies throughout the implementation effort
S13	Able to test the product in all aspects and able to support the development and execution of test scenarios
S14	Able to implement technologies into test and production environments
S15	Ability to Coach team members in specialized process skill sets if required

Each employee Skill score is calculated according to five principles introduced by Ahilton Barreto et al, [12]. Based on the working hours, the salary of each employee is calculated by Eqn (1). In the present study, 7 Regular employees and 3 Temporary employees are considered.

Table3. Employee wise salary based on working hours.

Emp ID	Basic Salary	Normal Salary	Overtime Salary	Legal Working Hours	Max Working Hours	Dedicated Hours	Gross Salary	Experience
E1	3000	100	150	180	220	170	22000	4
E2	2500	80	120	150	200	170	16900	3
TE1	3228	72	98	170	227	192	17624	8
TE2	4738	168	194	168	218	152	30274	8
TE3	5247	172	190	167	216	174	35301	10
E3	4247	168	196	172	196	152	29783	7
E4	2647	123	147	174	212	184	25519	6
E5	2894	142	184	185	226	154	24762	6
E6	4832	153	164	183	204	173	31301	5
E7	3854	146	185	138	184	172	30292	6

Considering salary of each employee, the proficiency of each employee is calculated according to Eqn (2) using values obtained from Table3.

Table4. Proficiency of employees

Task ID	E1	E2	TE1	TE2	TE3	E3	E4	E5	E6	E7
T1	4.4002	3.3802	3.5250	6.0550	7.0604	5.9568	5.1040	4.9526	6.2604	6.0586
T2	9.6800	5.7122	6.2121	1.8330	2.4923	1.7740	1.3024	1.2661	1.9595	1.8352
T3	2.1296	9.6536	1.0948	5.5493	8.7981	5.2836	3.3236	3.0365	6.1334	5.5592
T4	4.6851	1.6314	1.9295	1.6799	3.1058	1.5736	8.4817	7.5192	1.9198	1.6839
T5	1.0307	2.7571	3.4005	5.0860	1.0963	4.6867	2.1644	1.8619	6.0092	5.1011
T6	2.2675	4.6596	5.9931	1.5397	3.8703	1.3958	5.5234	4.6104	1.8809	1.5452
T7	4.9887	7.4787	1.0562	4.6614	1.3662	4.1572	1.4095	1.1416	5.8875	4.6808
T8	1.0975	1.3308	1.8615	1.4111	4.8230	1.2381	3.5969	2.8269	1.8428	1.4179
T9	2.4145	2.2491	3.2807	4.2722	1.7026	3.6876	9.1791	7.0000	5.7683	4.2951
T10	5.3119	3.8009	5.7819	1.2933	6.0103	1.0982	2.3424	1.7333	1.8055	1.3010

The proficiency values obtained from Table4, the fitness values of each employee are calculated based on the proficiency values using the equation (3)

Table5. Fitness values of employees

	Fitness									
Emp ID	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**
E1	0.345	0.463	1.075	0.281	2.910	0.327	1.273	0.921	1.827	0.219
E2	0.128	0.712	0.371	0.893	0.482	1.384	0.629	1.238	0.323	1.238
TE1	0.961	1.032	0.273	0.192	0.103	0.912	3.133	0.713	0.692	2.813
TE2	0.473	0.218	0.197	1.819	1.028	0.182	1.382	0.129	0.731	0.913
TE3	0.647	0.466	1.574	0.375	0.729	0.278	2.288	0.527	0.946	0.284
E3	0.174	1.539	0.278	1.588	0.686	1.732	0.733	0.276	1.388	0.293
E4	0.284	0.824	0.139	0.932	1.297	1.282	0.823	0.736	0.923	0.173
E5	0.216	2.534	1.952	0.948	0.483	0.373	0.593	0.977	0.189	0.994
E6	0.836	0.631	0.843	0.855	0.337	0.742	0.952	0.496	1.965	0.731
E7	0.728	1.097	0.973	0.136	1.972	0.632	0.862	0.176	1.273	0.782

To select an employee, the skill rating should be from (1.0-7.0) according to the five principles introduced by Ahilton Barreto et al, [4]. This is calculated based on the Eqn (3) & (4). When the skill rate is between (1.0-2.0), the employee is not suitable for the task. Whereas the employee having the skill rate between (3.0-5.0) may or may not be suitable for the task and when an employee skill rate is between (6.0-7.0), then it is stated as that employee is more suitable for that particular task.

After calculating the proficiency of an employee, human resource allocation is the next task. For this work, employee has to be assigned to high priority task. Here comes the task list with an $eligibleSet$ that contains all the tasks. Each task contains some weight on their edges in Task Precedence Graph (TPG). The construction of task list is built on the minimum slack (MINSLK) heuristic values obtained using the Eqn. (7) and (8). Another $eligibleSet$ is created for adding the tasks that are already selected and assigned to employee. When the task in first $eligibleSet$ is selected, the task is added to second $eligibleSet$ and removed from first set. This process is continued until all the tasks in first $eligibleSet$ become empty. The Task Precedence Graph (TPG) is constructed using the values of Table6.

Table6. Heuristic Values of each Task

Task ID	**Heuristic Value**
T1	1.56
T2	3.53
T3	3.31
T4	2.62
T5	1.09
T6	4.18
T7	5.30
T8	0.95
T9	2.97
T10	3.26

From the values obtained from Table6, the task with high heuristic value is having more priority and selected to place in the Task Precedence Graph.

VI. RESULTS

Each employee's heuristic value is calculated using Eqn (9), (10), (11) to find the suitable person for a particular task. The highest proficiency value and low salary will say that the employee is most suitable for that task. In this approach, the employee is placed in Eligible Set for selecting the employee for the task and after each selection; the employee will be removed from the Eligible Set after assigning the working hours. According to [18], the employee is selected by efficiency; the efficiency rate will be evaluated by

$$Efficiency = 2.94 \times EAF.(Size)^{E} \quad (12)$$

Table7. Employee Individual efficiency values

Task ID	**E1**	**E2**	**TE1**	**TE2**	**TE3**	**E3**	**E4**	**E5**	**E6**	**E7**
T1	28.97163	72.89735	26.82667	28.98787	44.98235	31.98301	53.13927	84.63281	87.81465	28.82665
T2	32.18987	44.49761	38.92783	72.92782	28.29743	36.79687	74.04188	72.18683	27.97467	76.19372
T3	73.92734	63.99248	33.78673	26.15737	33.97264	41.64099	27.87284	86.78628	39.43395	38.73846
T4	52.99277	29.97846	79.57514	38.23192	35.97247	46.51215	45.46269	26.979748	30.47760	47.29624
T5	29.98347	82.88264	37.92423	52.82831	45.87284	39.62332	39.97246	57.88728	43.97826	42.88649
T6	35.97724	64.88734	58.97867	56.98787	92.89748	33.24683	29.92748	28.87284	28.97788	45.92649
T7	91.15757	58.97836	66.78264	52.55504	36.82424	37.87827	28.98972	39.97248	29.41786	62.39447
T8	67.98737	35.09448	68.27927	81.78738	29.87257	53.28497	63.98927	67.60674	33.65627	63.26562

T9	81.78758	26.49528	28.78364	27.20352	47.87828	27.38917	82.89735	57.99549	38.00580	52.82472
T10	38.21643	43.66044	26.97387	71.78784	37.97242	79.137972	36.79687	68.97274	77.8255	57.41356

Based on the values obtained in Table6, the employee efficiency values can be used to identify in which task the employee is having high efficiency. From the values obtained in Table6, the following graphs are generated:

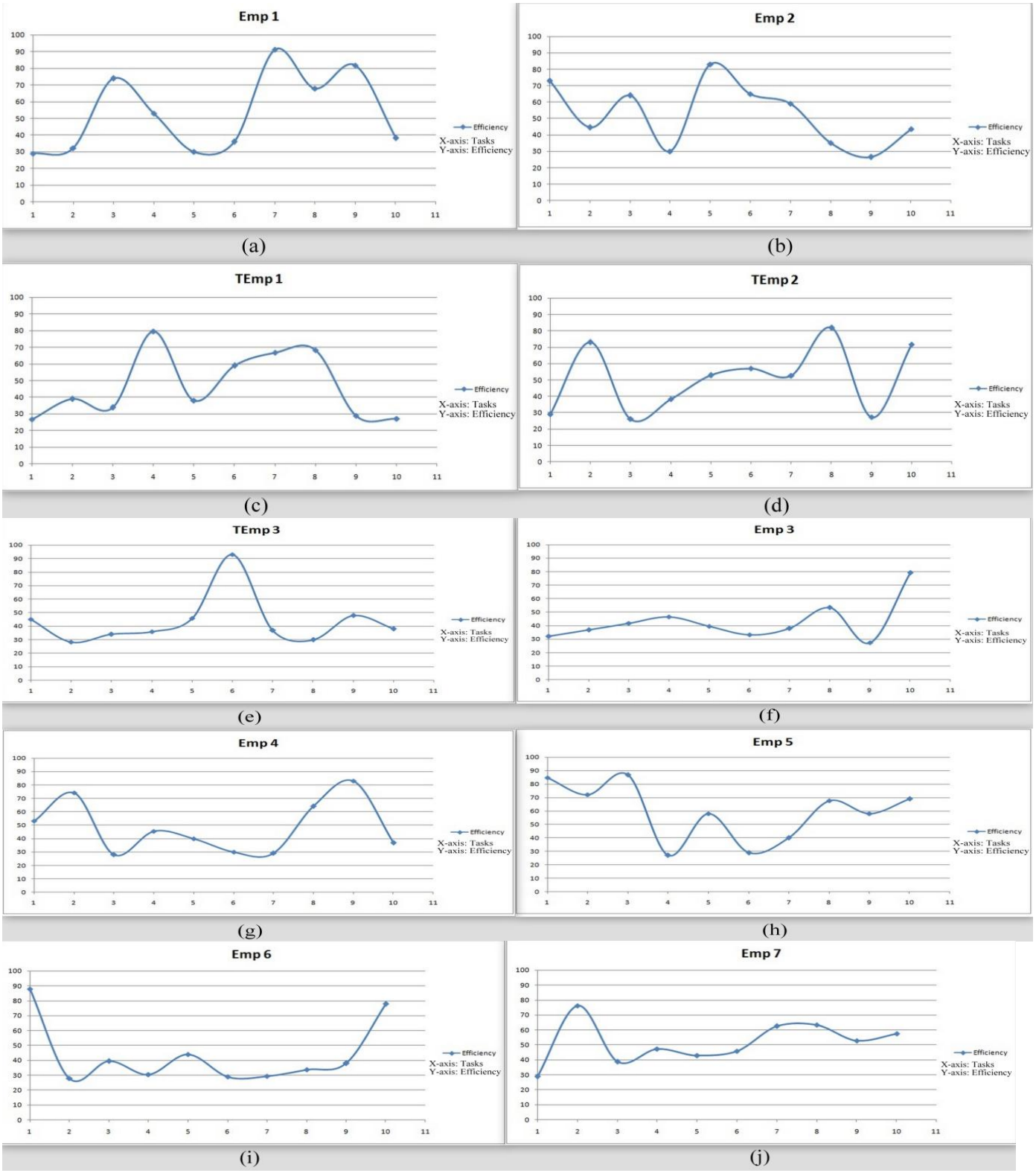

Fig1. Each Employee individual efficiency graphs

Fig1(a) – 1(j) shows individual Employee efficiency for all tasks, which are calculated using the Eqn (12).When allocating employee's to tasks, there comes a situation that has to select an employee who has low efficiency value. In that situation an optimistic decision has to take based on the Ant Colony Optimization approach. Every employee will not have the

skills to work with all tasks. Some may be fresher's, some may be well experienced, and some may be average experience in all aspects. This is the difficult task of selecting an employee. Here, the employee efficiency rate will help to solve the problem in a simple way. The efficiency scoring of all employees will be cross checked with the help of summation rule [14]. The high efficiency value of an employee of a task t_j will be allocated to that task. Through this proposed approach, the employees can be allocated to all tasks in a simple way with flexibility in work.

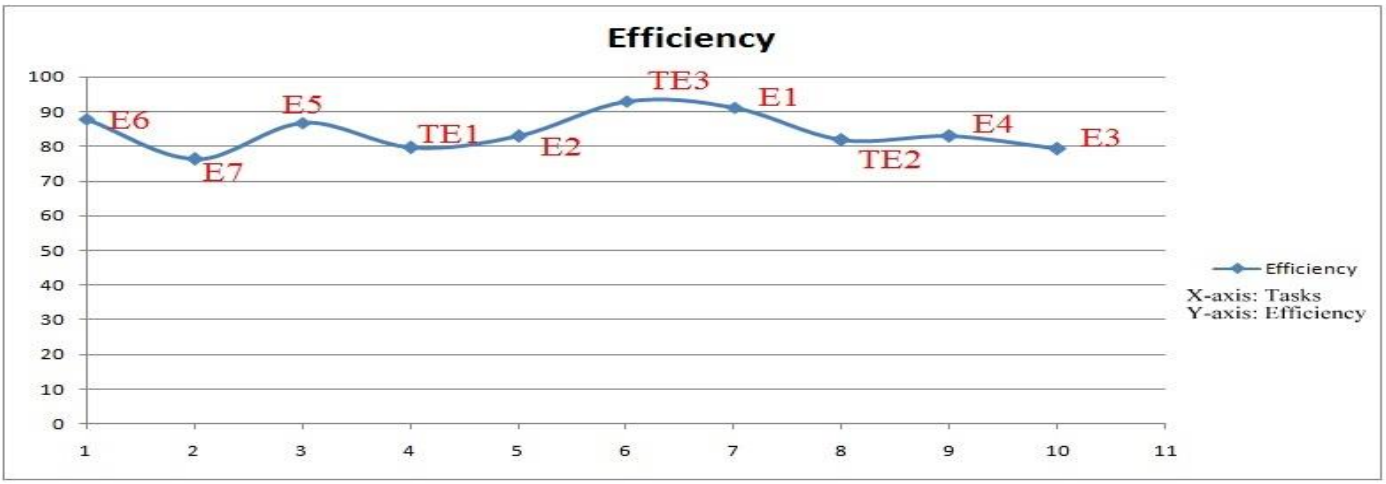

Fig2. Employee's allocation of tasks

VII. CONCLUSIONS

In this work, the efficiency of each individual employee is calculated using COCOMO II cost estimation. EBS is used to identify different events in project scheduling and planned working hours are computed, which are used in ACO for employee salary calculation and allocation of employees for various tasks based on their efficiency automatically. This technique will reduce cost and an automatically readjusts the tasks which cannot be completed within the given deadline based on the Task Precedence Graph and also identifies the priority of tasks to adjust Task Precedence Graph for task accomplishment.

REFERENCES

[1] C.K. Chang, H. Jiang, D. Zhu, Y. Di, and Y. Ge, "Time-line Based Model for Software Project Scheduling with Genetic Algorithms", Information and Software Technology, vol. 50, pp. 1142-1154, 2008.
[2] C.K. Chang, C. Chao, M.J. Christensen, and T.T. Nguyen, "Software Project Management Net: A New Methodology on Software Management", Proc. 22nd Ann. Int'l Computer Software and Applications Conf., 1998.
[3] T. Stutzle and H. Hoos, "Max-Min Ant System", Future Generation Computer Systems, vol. 16, no. 8, pp. 889-914, 2000.
[4] Wei-Neng Chen, "Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler", IEEE Transactions on Software Engineering, vol 39, no. 1, 2013.
[5] P. Brucker, A. Drexl, K. Neumann, R. Mohring, E. Pesch, "Resource Constrained Project Scheduling: Notation Classification, Models and Methods", European J. Operational Research, vol. 112, pp. 3-41,1999.
[6] M. Dorigo, A.Colorni, "Ant System: Optimization by a Colony of Cooperating Agents", IEEE Transactions on Systems Managements, and Cybernetics- Part B: Cybernetics, vol- 26, no. 1, pp. 29-41, 1996.
[7] M. Dorigo and L.M. Gambradella, "Ant Colony System: A Cooperative Learning Approach to TSP", IEEE Transactions on Evolutioary Computation, vol. 1, no. 1, pp. 53-66, 1997.
[8] M. Dorigo, "Ant Colony Optimization", MIT Press, 2004.
[9] L. Ozdamar, "A Genetic Algorithm Approach to a General Category Project Scheduling Problem", IEEE Transactions on Systems Management, and Cybernetics- Part C: Applications and Rev., vol. 29, no. 1, pp. 44-59, 1999.
[10] S. Hatmann and R. Kolisch, "Heuristic Algorithms for Solveing the Resource- Constrained Project Scheduling Problem: Classification and Computational Analysis", Handbook on Recent Advances in Project Scheduling, J. Weglarz., ed., pp. 197-212, Kluwer, 1999.
[11] C.K. Chang, T.Zhang "Genetic Algorithms for Project Management", Annals of Software Engineering, vol. 11, pp. 107-139, 2001.
[12] A. Barreto, C.M.L. Werner, "Staffing a Software Project: A Constraint Satisfation and Optimization-Based approach", Computers and Operations Research, vol. 25, pp. 3073-3089, 2008.
[13] C. Blum, "An Ant Colony Optimization Algorithm for Shop Scheduling Problems", J. Math. Modeling and Algorithms, vol. 3, pp. 285-308, 2004.
[14] C. Blum, "Beam-ACO-Hrbridizing Ant Colony Optimization with Beam Search: An Application to Open Shop Scheduling", Computers and Operations Research, vol. 32, pp.1565-1591, 2005.
[15] J.S. Aguilar-Ruiz, I. Ramos, J.C. Riquelme and M. Toro, "An Evolutionary Approach to Estimating Software Development Projects", Information and Software Technology, vol. 43, pp. 875-882, 2001.
[16] A. Shtub, J.F. Bard, and S. Globerson, Project Management: Processes, Methedologies and Economics, Second ed. Prentice Hall, 2005.

[17] A. Drexl, P. Brucker, K. Neumann, E. Pesch, "Resource Constrained Project Scheduling: Notation, Classification, Models and Methods", European J. Operational Research, vol. 112, pp. 3-41, 1999.
[18] Alaa F. Sheta and Sultan Aljahdali, "Software Effort Estimation Inspired by COCOMO and FP Models: A Fuzzy Logic Approach", International Journal of Advanced Computer Science and Applications, vol. 4, no. 11, 2013.