International Journal on Applicationsin Information and Communication Engineering
Volume 2:Issue 6: June 2016, pp 17-20. www.aetsjournal.com ISSN (Online) : 2394 - 6237

Software Project Scheduling Methodologies - Overview

Sarojini Yarramsetti, Renisha P.S

***Abstract*—** In software Project Planning, task scheduling and human resource allocation to the task plays a vital role. Different software project tasks require employees with different skills and skill proficiency of employees influences the efficiency of project execution. Similarly assigning the employees to the best fitted tasks results in high quality of project production and reduces the construction cost. For scheduling and staffing, there are several software projects management methods have been developed. This paper presents an analysis of some of these software project scheduling techniques which are used to address a solution for scheduling and staffing. Uncertainty is an important concern in project scheduling since it can cause infeasibilities and construction disturbances. Thus scheduling under uncertainty has received a lot of attention in recent years. The purpose of this paper is to review the main methodologies that have been developed to address the problem of Software Project Scheduling and uncertainty as well as to identify the main challenges in this area. The uncertainties in process scheduling are first analyzed, and the different mathematical approaches that exist to describe process uncertainties are classified. Based on the different descriptions for the uncertainties, alternative scheduling approaches and relevant optimization models are reviewed and discussed. Further research challenges in the field of process scheduling under uncertainty are identified and some new ideas are discussed.

***Keywords*—** Ant Colony Optimization (ACO), Event Based Scheduler (EBS), Particle Swarm Optimization (PSO)

I. INTRODUCTION

Software Project planning is a process which is performed before the construction of software actually starts. Companies have to make efficient project plans to reduce the cost of software construction.[1]. To plan software project the project manager needs to estimate the project manager needs to estimate the project workload and cost and decide the project schedule and resource allocation. Famous model COCOMO[2] widely used for workload and cost estimation. Management is usually used project management tools and techniques for scheduling and staffing.

A software project is a people – intensive activity and its related resources are mainly human resources [3] . Different software project tasks require different skill employees and skill proficiency of employees influences the efficiency of project execution. Similarly assigning employees to the best fitted tasks results in high quality of software production. Therefore, task scheduling and resource allocation play a vital role in software project plan.

Various software project tasks such as cost estimation [4], module clustering [5], design [6], buildingtesting [7] and software release planning [8] have been modeled as search based problems and metaheuristic algorithms have been applied successfully. Search –based approaches are promising way for software project planning. This paper will presents an analysis on various available search based approaches like GA,SPMnet,PMnet,RCPSP, Tabu Search, ACO with EBS.

Table 1: Various Attributes considered for each and every Employee in a Company.

Sl.no.	Attributes	Description
1	b_{s_i}	The basic salary for the employee per time period (e.g., month).
2	h_{s_i}	The salary for the employee's per-hour normal work.
3	oh_{s_i}	The salary for the employee's per-hour overtime work.
4	nh	Legal normal working hours per month,
5	$maxh_i$	Maximum possible working hours per month of the employee for the project.
6	$[join_i; leave_i]$	The time window when the employee is available for the project.
7	$\{s_i^1, s_i^2 \ldots . s_i^{\Phi}\}$	The skill list for the employee, where Φ is the number of skills and $s_i^j \in [0,5]$is the proficiency score of the j^{th} skill. $s_i^j = 0$ Means the employee does not have the skill and $s_i^j = 5$ means the employee is masterly on that skill.
8	$Taskexp_e$	Employee's experience level.

Uncertainty is an important concern in project scheduling since it can cause infeasibilities and construction disturbances. Thus scheduling under uncertainty has received a lot of attention in recent years. The purpose of this paper is to review the main methodologies that have been developed to

Sarojini Yarramsetti, Assistant Professor/ Department of IT , Hindusthan College of Engineering and Technology- Coimbatore-32
Renisha P.S, Assistant Professor/ Department of IT ,Hindusthan College of Engineering and Technology- Coimbatore-32

address the problem of Software Project Scheduling and uncertainty as well as to identify the main challenges in this area. The uncertainties in process scheduling are first analyzed, and the different mathematical approaches that exist to describe process uncertainties are classified. Based on the different descriptions for the uncertainties, alternative scheduling approaches and relevant optimization models are reviewed and discussed. Further research challenges in the field of process scheduling under uncertainty are identified and some new ideas are discussed.

II. Representation And The Event-Basedscheduler

The EBS is characterized by making new assignments at events. We regard the time t as an event if t satisfies any oneof the following three conditions: 1) 1 is the beginning of the project, 2) any employee joins or leaves the project at time t,or 3) any task just finished in the previous time period and the corresponding resources become released and availableat t. The EBS adjusts a plan into an actual timetable by two rules. First, if there is resource conflictbetween two tasks, the task that appears earlier in the task list has a higher priority to use the resource. That is,Assuming that the ith employee is originally planned to simultaneously dedicate pwhij and pwhik of his workinghours to tj and tk, respectively, if pwhij þ pwhik>maxhi, the employee will first dedicate his working hours to thetask with a higher priority. Second, new workload assignments are only made when events occur. If no employeesjoin or leave the project or no human resource is released by the tasks just finished, the workload assignments remainthe same as the previous time period. The pseudo code of the EBS is given in Fig. 2. At the beginning, the start time of the project and the time when employees join or leave the project are set as events. Then the scheduler assigns workloads in chronological order with the growing of time t. If t is an event, the scheduler reassigns the actual workloads according to the priority of a task defined by the task list and the originally planned workloads (lines 6-16). Otherwise, if t is not an event, the workloads at t remain the same as those at t _ 1 (line 19). After allocating the actual workloads at t, the achievement Af of vj at t is calculated by (line 21). If there is any task finished at t, the time t + 1 is set as an event because the resources assigned to these tasks can all be released at t + 1. These steps run repeatedly until all tasks of the project have been finished. In Fig. 2, there are local refinement steps in lines 17 and 24.

EBS composed of task list and employee allocation matrix so it address the both the problems of task scheduling and employee allocation. The main goal of EBS is to adjust the allocation of employee whenever events occur and keep it constant when events do not take place.

The following condition is determined as events :

1) Starting of the project
2) Resource released by completed task
3) Employee enters and leaves the project.

EBS, Schedules the tasks to the available resources to complete the projects.

```
procedure scheduler //EBS
01  initialize the number of available human resources during the whole scheduling window;
02  set the beginning time and the time when employees join or leave the project as events;
03  t = 1;
04  while the project is not finished
05      if t is an event
06          find the tasks that can be implemented at time t and arrange these tasks into a sequence seq
              according to the order defined by the task list;
07          while seq is not empty
08              set tj as the first task in seq and remove tj from seq;
09              for i=1 to m                //for every employee
10                  if the planned working hours pwhij is not larger than the remaining available working
                      hours of the i-th employee at t
11                      whij^t = pwhij;
12                  else
13                      whij^t is set to the remaining available working hours of the i-th employee at t;
14                  end if-else
15              end for
16          end while
17          (local refinement: let regular employees devote all of their normal working hours to the project.)
18      else
19          the workload assignments are the same as those at t-1;
20      end if-else
21      evaluate the completion situation of the tasks at time t;
22      if there are any tasks finished at time t
23          set the time t+1 as an event;
24          (local refinement: release redundant working hours of employees.)
25      end if
26      t = t+1;
27  end while
end procedure
```

Fig. 2.Pseudo code of the EBS.

III. An Overview Of Particle Swarm Optimization

PSO is originally attributed to Kennedy, Eberhart and Shi[9][10] and was first intended for simulatingsocial behavior,[11] as a stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization.

A. Particle Swarm Optimizationalgorithm

A variant of the PSO algorithm works on population called a swarm of candidate solutions called particles. These particles are moved around in the search-space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position. When improved positions are being discovered these will then come to guide the movements of the swarm. The process is repeated and by doing so it is hoped, but not guaranteed, that a satisfactory solution will eventually be discovered.

Formally, let $f: \mathbb{R}^n \to \mathbb{R}$be the cost function which must be minimized. The function takes a candidate solution as argument in the form of a vector of real numbers and produces a real number as output which indicates the objective function value of the given candidate solution. The gradientof f is not known. The goal is to find a solution $\mathbf{a}$ for which $f(\mathbf{a}) \leq f(\mathbf{b})$ for all $\mathbf{b}$ in the search-space, which would mean $\mathbf{a}$ is the global minimum. Maximization can be performed by considering the function $h = -f$ instead.

Let S be the number of particles in the swarm, each having a position $\mathbf{x}_i \in \mathbb{R}^n$ in the search-space and a velocity $\mathbf{v}_i \in \mathbb{R}^n$. Let $\mathbf{p}_i$ be the best known position of particle i and let $\mathbf{g}$ be the best known position of the entire swarm. A basic PSO algorithm is then:[7]

- For each particle $i = 1, ..., S$ do:
  - Initialize the particle's position with a uniformly distributed random vector: $\mathbf{x}_i \sim U(\mathbf{b}_{lo}, \mathbf{b}_{up})$, where $\mathbf{b}_{lo}$ and $\mathbf{b}_{up}$ are the lower and upper boundaries of the search-space.
  - Initialize the particle's best known position to its initial position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
  - If ($f(\mathbf{p}_i) < f(\mathbf{g})$) update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
  - Initialize the particle's velocity: $\mathbf{v}_i \sim U(-|\mathbf{b}_{up}-\mathbf{b}_{lo}|, |\mathbf{b}_{up}-\mathbf{b}_{lo}|)$
- Until a termination criterion is met (e.g. number of iterations performed, or a solution with adequate objective function value is found), repeat:
  - For each particle $i = 1, ..., S$ do:
    - For each dimension $d = 1, ..., n$ do:
      - Pick random numbers: $r_p, r_g \sim U(0,1)$
      - Update the particle's velocity: $\mathbf{v}_{i,d} \leftarrow \omega\, \mathbf{v}_{i,d} + \varphi_p r_p\, (\mathbf{p}_{i,d}-\mathbf{x}_{i,d}) + \varphi_g r_g\, (\mathbf{g}_d-\mathbf{x}_{i,d})$
    - Update the particle's position: $\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i$
    - If ($f(\mathbf{x}_i) < f(\mathbf{p}_i)$) do:
      - Update the particle's best known position: $\mathbf{p}_i \leftarrow \mathbf{x}_i$
      - If ($f(\mathbf{p}_i) < f(\mathbf{g})$) update the swarm's best known position: $\mathbf{g} \leftarrow \mathbf{p}_i$
- Now $\mathbf{g}$ holds the best found solution.

The parameters ω, φ_p, and φ_g are selected by the practitioner and control the behavior and efficacy of the PSO method.

IV. RESULT AND DISCUSSIONS

The performance of the optimization algorithmsare evaluated based on the parameters like

- Convergence behavior
- Processing Time

A. Convergence Behavior

Table 2 shows the performance comparison of the optimization techniques such as Genetic Algorithm, ACO-EBS and the PSO-EBS.

The PSO-EBS optimization algorithm like, PSO_GA, PSO_ABC etc., outperforms the GA Algorithm and ACO algorithm in attaining the reliability in terms of convergence behavior.

Figure 1 shows the comparison of the convergence behavior of the GA, ACO and the PSO approach. It is observed from the figure that the PSO converges in lesser iterations when compared with the other optimization techniques. For instance, the GA approach takes 90 iterations for convergence, ACO approach takes 70 iterations whereas the hybrid optimization approach takes 40 iterations for convergence, thus the PSO optimization technique is very significant when compared with the other optimization approaches taken for consideration.

B. Processing Time

Table 2 shows the performance comparison of the optimization techniques such as Genetic Algorithm, ACOs and the PSO optimization approach.

The PSO optimization algorithm outperforms the GA and ACOs algorithm in attaining the reliability in terms of the processing time.

It is observed from the figure 3 that the PSO optimization technique takes processing time of 9 seconds, whereas the other optimization techniques such as GA and ACOs take longer processing time such as 20 and 17 seconds respectively.

The experiments are evaluated based on some other parameters like reliability, testing cost and the run time. In this section, we compare the hybrid optimization approach with traditional single-objective approaches. Table 3 shows that the comparison of various approaches. From the table, it shows that the hybrid optimization approach is more efficient in solving the problems of resource allocation in the software products etc., than the other existing approaches.

From the figures 1, 2 and 3 it is observed that the PSO optimization approach is more efficient than the other existing approaches.

Finally, it is concluded that the PSO optimization approaches gives better results in the field of software project planning and scheduling.

Table1: Performance Comparison of the Optimization Techniques using Convergence Behavior

NAME OF INSTANCES	GA	ACO-EBS	PSO-EBS
TASK1	3.5	3	2.5
TASK2	2.65	2.25	1.75
TASK3	2.8	2.6	2.1
TASK4	2.25	2.15	1.8
TASK5	2.55	2.2	1.15
TASK6	2.65	2.35	1.2
TASK7	2.45	2.5	1.5
TASK8	2.11	1.85	1.3
TASK9	1.5	1.25	1.1
TASK10	1.35	1.25	1.05

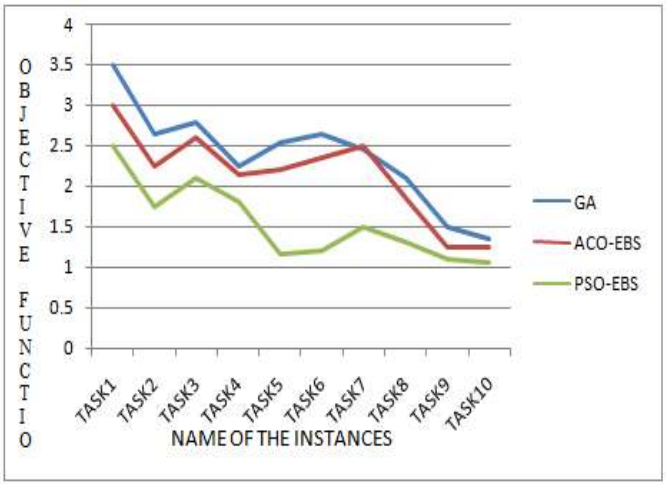

Figure 1: Comparison of Convergence Behavior

Table 2: Performance Comparison of the Optimization Techniques using Processing Time.

Optimization Algorithms	Processing Time (sec)
GA	20
ACO-EBS	17
PSO-EBS	9

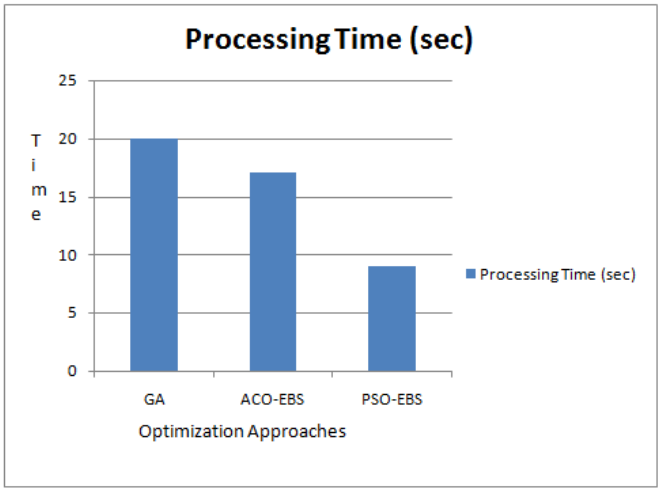

Figure 2: Comparison of Processing Time

Table 3: Comparison of various approaches

Approaches	Reliability	Testing cost (units)	Total Testing Resource consumed
GA	0.73	9.426	14.75
ACO-EBS	0.85	9.124	13.81
PSO-EBS	0.99	8.05	12.75

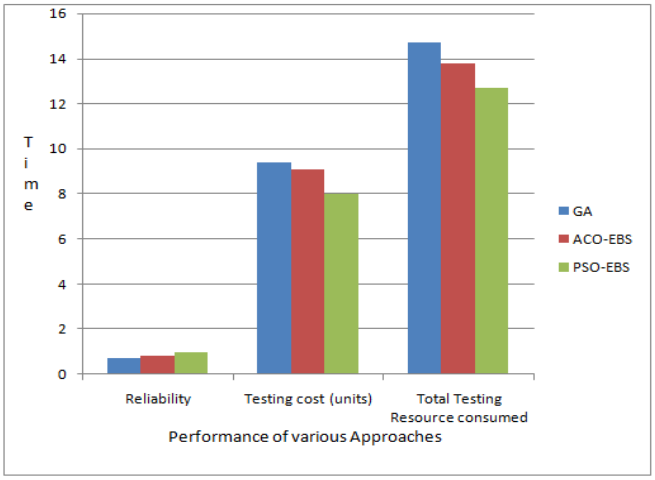

Figure 3: Comparison of Reliability, Cost, Resource consumed in various approaches

V.CONCLUSION

Software Project Planning is essential for quality software construction.Efficient software project planning will reduces the construction cost and time. Various tasks such as Analysis , design,build,test should be scheduled correctly. Assigning right task to the right person of human resource is also important, which increases the quality of product. This pare present various search based approaches for software project planning.Full Automated tool for scheduling and resource allocation is challenging in software engineering.

REFERENCES

[1] N. Nan and D.E. Harter, "Impact of Budget and Schedule Pressure on Software Development Cycle Time and Effort," IEEE Trans. Software Eng., vol. 35, no. 5, pp. 624-637, Sept./Oct. 2009.
[2] B. Boehm et al., Software Cost Estimation with COCOMO II. Prentice-Hall, 2000.
[3] A. Barreto, M. de O. Barros, C.M.L. Werner, "Staffing a Software Project: A Constraint Satisfaction and Optimization-Based ap-proach," Computers & Operations Research, vol. 35, pp. 3073-3089, 2008.
[4] 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.
[5] K. Praditwong, M. Harman, and X. Yao, "Software Module Clustering as a Multi-Objective Search Problem," IEEE Trans. Software Eng., vol. 37, no. 2, pp. 264-282, Mar./Apr. 2011.
[6] C.L. Simons, I.C. Parmee, and R. Gwynllyw, "Interactive, Evolutionary Search in Upstream Object-Oriented Class Design,"IEEE Trans. Software Eng., vol. 36, no. 6, pp. 798-816, Nov./Dec. 2010.
[7] M. Harman and P. McMinn, "A Theoretical and Empirical Study of Search-Based Testing: Local, Global, and Hybrid Search," IEEE Trans. Software Eng., vol. 36, no. 2, pp. 226-247, Mar./Apr. 2010.
[8] V. Garousi, "A Genetic Algorithm-Based Stress Test Requirements Generator Tool and Its Empirical Evaluation," IEEE Trans. Software Eng., vol. 36, no. 6, pp. 778-797, Nov./Dec. 2010.
[9] Kennedy, J.; Eberhart, R. (1995). "Particle Swarm Optimization".Proceedings of IEEE International Conference on Neural Networks. pp. 1942–1948. doi:10.1109/ICNN.1995.488968.
[10] Shi, Y.; Eberhart, R.C. (1998). "A modified particle swarm optimizer". Proceedings of IEEE International Conference on Evolutionary Computation. pp. 69–73.
[11] Kennedy, J. (1997). "The particle swarm: social adaptation of knowledge". Proceedings of IEEE International Conference on Evolutionary Computation. pp. 303–308.