A Hybrid Approach for Software Project Scheduling

V.Karthiga[1] and K.Sumangala[2]

[1]M.Phil Scholor, Department of Computer Science,
Vellalar College for women (Autonomous), Erode-12, Tamil Nadu, India.
[2]Assistant Professor, Department of Computer Applications,
Vellalar College for women (Autonomous), Erode-12, Tamil Nadu, India.

ABSTRACT

This research work handles Software project scheduling problem with a hybrid approach. In this, a hybrid approach can handle combination of two different algorithms they are Event Based Scheduler(EBS) and Ant Colony Optimization(ACO). Which provide scheduling and also allocate the resources based on the project. The basic idea of the EBS is to adjust the allocation of employees at events and keep the allocation unchanged at non-events. ACO is to assign the project tasks to suitable employees with required skills.

Keywords: Event Based Scheduler(EBS), Software project planning, Snap Scheduler, Ant Colony Optimization(ACO).

1. INTRODUCTION

To develop a software project, the project manager needs to estimate the project workload and cost and decide the project schedule and resource allocation. Software project tasks require employees with different skills, and skill proficiency of employees significantly influences the efficiency of project execution. Assigning employees to the best-fitted tasks is challenging for software project managers, and human resource allocation has become a crucial part in software project planning based on the skills and experiences of the employees.

Project management techniques usually regard task scheduling and human resource allocation as two separated activities and leave the job of human resource allocation to be done by project managers manually, resulting in inefficient resource allocation and poor management performance. Main resources in software development are humans instead of big machines, resources in software projects can usually be allocated in a more flexible way than those in construction or manufacturing projects.

The task list defines the priorities of tasks to consume resources, and the planned employee allocation matrix specifies the originally-planned workload assignments. The EBS[10] regards the beginning time of the project, the time when resources are released from any finished task, and the time when employees join or leave the project as events. To generate an actual timetable, the EBS adjusts the workload assignments of employees at events and resource conflict is solved according to the priority defined by the task list.

ACO builds solutions in a step-by-step manner and enables the use of problem-based heuristics to guide the search direction of ants, it is possible to design useful heuristics to direct the ants to schedule the critical tasks as early as possible and to assign the project tasks to suitable employees with required skills. Therefore, ACO is promising to converge fast and perform well on the considered problem.

2. METHODOLOGY

Ant Colony Optimization (ACO) algorithm to solve instances of the software project scheduling problem by taking employee experience and training model. This approach is characterized by the local optimization techniques: Two kinds of cluster information (employee experience and training programmes) in path level and attribute level.

The proposed model provides a flexible and effective way for managing human resources it is promising to apply the proposed approach to other complex human-centric projects like consulting projects.

Proposes a new enhanced Ant Colony Optimization approach in which class and attributes (like employee experience, training model with varying employee count (by taking admission and relieving during the project period)) are of dynamic nature. At the same time, attribute level clustering and finding best ant in attribute level is also derived.

Furthermore, the architecture (e.g., infrastructure) is capable of evaluating the project scheduling and thereby satisfies the ever-changing requirements of the software in convenience. Whereas the traditional service mission is “can do it” (i.e., provide the optimal service within the existing service architecture), the current service mission is “do it best” (i.e., provide the optimal service within an optimized service architecture).

In comparison with the existing traditional ant colony optimization techniques, two important extensions are addressed in this project. First extension considers the dynamic class levels. Second extension deals with the dynamic attribute levels. It assures that both the class and the attributes are variable.

Software development is a people-intensive activity. To manage employees, an employee database is needed to record the employees’ information of wages, skills and working constraints. The problem of employee allocation is to assign employees to suitable tasks so that the tasks can be done efficiently. Suppose **m** employees are involved in the project, for the i-th employee (i=1,2,…,m), the attributes are the basic salary for the employee per time period (e.g., month) (bs_i), the salary for the employee’s per-hour normal work (hs_i), the salary for the employee’s per-hour overtime work (ohs_i), legal normal working hours per month (n_h), maximum possible working hours per month of the employee for the project($maxh_i$), the time window when the employee is available for the project([joini, leavei]), the skill list for the employee, where Φ is the number of skills($\{s^1, s^2, .. \square ., S^{\Phi}\}$)

and $S_i^j \in [0,5]$ is the proficiency score of the j-th skill. Here the skills can be documenting, C++ programming, GUI design and any other technical abilities. $S_i^j = 0$ means the employee does not have the skill and $S_i^j = 5$ means the employee is masterly on that skill.

Basic salaries are paid to regular employees every month regardless of the workloads the employees devote to the project. On the other hand, per-hour normal work salaries are paid according to the working hours. Usually, there are two types of employees: regular ones and temporary ones. Regular employees have stable basic salaries, while temporary ones do not have basic salaries but have much higher per-hour working salaries. The software project planning problem involves task scheduling and employee allocation a plan for a project must specify when the tasks of the project are processed and how the workloads of employees are assigned to the tasks. More specifically, the plan has to determine the start time $start_j$ and the finish time $finish_j$ of each task t_j ($j \in \{1,2,\ldots,n\}$), and the working hours of all employees wh_{ij}^t of all employees $i \in \{1,2,\ldots,m\}$ to the task t_j during the time window $t \in [start_j, finish_j]$.

The salary [10] for the i-th employee at the t-th month is calculated by

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

The EBS is characterized by making new assignments at events. This regard the time *t* as an event if *t* satisfies any one of the following three conditions: 1) *t*=1 is the beginning of the project, 2) any employee joins or leaves the project at *t*; or 3) any task just finished in the previous time period and the corresponding resources become released and available at *t*.

Fig 1. Steps of EBS

Step 1: Initialize the number of available human resources

Step 2: Find the task

Step 3: If the planned working hours is not greater than the remaining working hours of the i-th employee, assign planned working hours of the project to the number of working hours of the i-th employee for the task j

Step 4: Else, the number of working hours of the i-th employee for the task j is set to the remaining working hours of the i-th employee at t.

Step 5: Evaluate the completion situation of the task at time t

Step 6: If any task is finished at time t, set t+1 as event

Step 7: Increment t

Then to solve the software project planning problem, ACO is used.

Step 1: Initialize the number of employee(i) and assign i=1

Step 2: The i-th ant contructs a task list and build an employee allocation matrix

Step 3: Update local pheromone

Step 4: Convert the solution into timetable using EBS

Step 5: Increment i

Step 6: Update Global Pheromone

Step 7: Return the best solution

Fig 2. Steps of ACO

3. RESULTS AND DISCUSSIONS

Experimental analysis is indented to be of use to researchers from all fields who want to study algorithms experimentally. It has a goal to provide a useful guide to new experimental list about how such work has made.

3.1.TEST OBSERVATION DATA

The experiment is conducted based on total hours of the project and number of employees and also the minimum and maximum workings hours to the completion for work.

Table 1. Experimental result based on total working hours and number of employees of the project.

Project Hours	Employees	Tasks	Min Days	Max Days
300	5	2	4	8
300	10	3	7	31
400	10	3	12	31
400	10	4	6	32
500	8	3	7	39
500	10	3	7	34
300	10	2	7	22
400	10	2	14	35
500	10	2	7	35
600	10	3	7	38

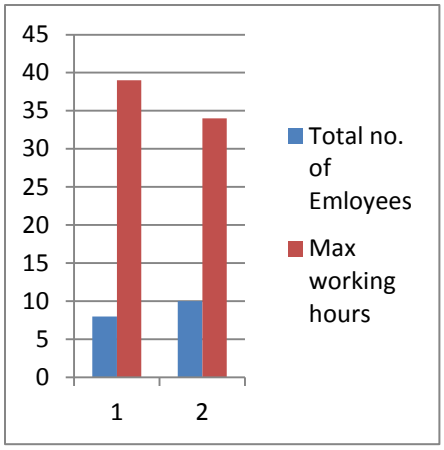

Fig 3 . Total number of Employees is compared with maximum working hours

The fig 3 gives the comparison of the total no. of Employees with the maximum no. of working hours for the project. Here fifth and sixth data field under the column heading employees shows the inverse proportionate of the working hours(WH) and the Employee count(EC)the condition.

$$EC \propto 1/WH$$

The x-axis shows the no. of dataset which helps to conclusion (two data fields are taken) and y-axis shows the working hours for the project.

Here Snap Scheduler is compared with Event Based Scheduler. The fig 4 shows the result of Snap Scheduler.

Fig 4. Snap Scheduler for employee allocation

The experiment is conducted based on Snap Scheduler algorithm with 48 working days and 9 employees. 48 working days are splitted into 8 sets. So each set contain 6 days, in that only 4 days are taken to work. Also employees are splitted into 3 sets. In the fig 4, on shows that working days of the employees and off shows that non-working days of the employees. Here the employees have to be in the project for 48 days.

The figure 5 shows the result of Event Based Scheduler.

Fig 5. Event Based Scheduler

The experiment is conducted based on Event Based Scheduler, with 5 employees and 2 tasks and shown in Fig 5. Here Employee 1 has given contribution to both the tasks. Employee 3 and 5 have contributed only in first task. So in the remaining tasks they may concentrate for doing some other project.

In Snap Scheduler, the employee can work only in working days while considering days, where in Event Based Scheduler (EBS), the employees can work either in working days or in non working days. Here days is not a matter and considered only hours at any days for the convenience of the employee.

4. CONCLUSION

The difficulty in staff scheduling is taken into consideration and so the Ant Colony Optimization technique is enhanced to solve the problem. The application satisfies the requirements by applying the algorithm for given POP size and global pheromone update count with given number of employees and tasks.

The employee count and task count are given different values and the result is derived. They can be viewed for various analysis purposes.

In future this algorithm can be applied to multiple projects will same employee count.

The new system is designed such that those enhancements can be integrated with current modules easily with less integration work.

5. REFERENCES

[1] C. Blum, "Beam-ACO – hybridizing ant colony optimization with beam search: an application to open shop scheduling," *Computers & Operations Research*, vol. 32, pp. 1565-1591, 2005.

[2] C. Blum and M. Sampels, "An ant colony optimization algorithm for shop scheduling problems," *Journal of Mathematical Modelling and Algorithms*, vol. 3, pp. 285-308, 2004.

[3] C.K. Chang and M. Christensen, "A net practice for software project management," *IEEE Software*, vol. 16, no. 6, pp. 80-88, 1999.

[4] W.-N. Chen and J.Zhang, "An ant colony optimization approach to a Grid workflow scheduling problem with various QoS requirements", IEEE Transactions on System, Man, and Cybernetics, Part C, vol.39, no.1, pp.29-43, 2009

[5] M. Dorigo, V. Maniezzo, A. Colorni,,"Ant system: optimization by a colony of cooperating agents," IEEE Transactions on Systems Man, and Cybernetics- part B: Cybernetics, vol.26, pp.29-41, 1996

[6] R.-G. Ding and X.-H. Jing, "Five principles of project management in software companies," Project Management Technology (in Chinese), vol.1, 2003.

[7] L.C. Liu and E. horowitz, "A formal model for software project management," IEEE Transactions on software Engineering, vol.15, no.10, pp. 1280-1293, 2001.

[8] D. Merkle, M. Middendorf, H. Schmeck, "Ant colony optimization for resourceconstrained project scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 333-346, 2002.

[9] N.Nan and D.E. Harter, "Impact of budget and schedule pressure on software development cycle time and effort," IEEE Transactions on Software Engineering, vol.5, no.5, pp.624-637, 2009.

[10] Wei-neng and Chen and Jun Zhang, "Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler".

AUTHOR'S PROFILE

V. Karthiga received the B.Sc Maths(C.A) from Bharathiar University in 2008 and M.C.A degrees in computer science from Anna University, Coimbatore in 2011 respectively. She is currently doing M.Phil. degree in Computer Science from Bharathiar University, Coimbatore.

Mrs. K. Sumangala completed B. Sc Computer Science & M.C.A in Madras University , Chennai. Doing Ph.D in Bharathiar University, Coimbatore. 18 years experience in handling classes for B.Sc Computer Science, Computer Application, IT, CT, M.Sc Computer Science and M.C.A. 8 years Research experience. Producing 12 M.PhilScholars. Currentlyguiding 3 M.PhilScholars.