A Jumper Based Ant Colony Optimization for Software Project Scheduling and Staffing with an Event-Based Scheduler

Nithya.G¹, Dhivy priya R², Harini S³, Menakapiya S⁴

¹²³⁴, Sri Krishna College of Engineering and Technology & Coimbatore, India

Abstract- Many computer aided techniques have been developed in recent years for managing the software projects which is challenging in the environment of software engineering. Measures in software project management involves project planning, project scheduling, risk management and managing people. The task of project scheduling and human resource allocation leads to poor management in most of the companies. Thus an effective model has to be developed to deal with both the allocation of human beings to various projects and to schedule them without restricting the flexibility. This paper puts forth a new concept of jumper based technique to concern with project scheduling and human resource allocation. To work with task lists and employee allocation matrix, an ant colony optimization algorithm is further designed and an event based scheduler (EBS) is used to deal with the events. The proposed approach enables an adequate modeling for software crisis and task preemption preserving the flexibility in human resource allocation.

Index Terms- ant colony optimization (ACO), Event based scheduler (EBS), Jumper, project scheduling, Risk management, resource allocation, Software project planning (SPP), workload assignment

I. INTRODUCTION

With the quickened development of the software companies, industries are now facing a heavy competitive market. Companies have to make accurate and efficient project plans to reduce the cost in the construction of software. On the other hand, in case of large-scale projects, project planning seems to be very complex. In China it was revealed that more than 50 percent of software projects failed because of inefficient planning of project tasks and human ability. These result in a growing demand for developing effective computer aided appliances for project planning due to the need and troubles in project sketching process in recent years. In case of workload and cost estimation, some famous models like CCOMO have been developed and widely used. For managing scheduling and staffing, similarly to other projects management is usually conducted by project management tools and techniques.

For example, project management techniques like the program evaluation and review technique (PERT), the critical path method (CPM), and the resource-constrained project scheduling problem (RCPSP) model have been applied in software project planning. Furthermore, as the main resources in software development are humans instead of machines, resources in software projects can usually be allocated in a more adjustable way than those in producing projects. If task preemption is accurately designed, availability of humans can be organized in a more adequate way. Presently, task preemption is still sparsely considered in project management models and tools. Many existing models have the guess that each employee can only be assigned to a single task at a time. This guess reduces the flexibility of resource allocation in project planning. Search-based approaches may also become a promising way for software project planning. Duggan et al. and Barreto et al. built models for the staffing problem of software projects and proposed genetic algorithm (GA) approaches. But, their models only focused on staffing and the problem of task scheduling was not considered. Chang et al. proposed the software project management net (SPMnet) model and the project management net (PM-net) model successively, and then further improved the models to a richer version with a GA. Bellenguez and Ne’ron proposed a multitask scheduling model by extending the traditional RCPSP model. The model considers both the problems of human resource allocation and task scheduling, and takes the skill proficiency of employees and resource conflict into consideration. Tabu search (TS), branch and bound, and GA have been developed for the model. In this paper, we develop a practical and effective approach for the task scheduling and human resource allocation problem in software project planning with an ant colony optimization (ACO) algorithm. Different from the existing approaches, the proposed method is characterized by the following two features. First, a representation scheme with event-based scheduler (EBS) is developed. The proposed scheme reduces the size of the search space and thus accelerates the search process.
ACO promises to assemble fast and perform well on the given problem. The rest of this paper is organized as follows:

1. Section 2 formulates the model for software project task scheduling and human resource allocation.
2. Section 3 reviews the related work of scheduling and staffing for software projects.
3. Section 4 introduces the representation scheme and the EBS.
4. In Section 5, the ACO algorithm with jumper concept is proposed.

II. MODEL DESCRIPTION

This paper considers real time problems such as human resource allocation and task scheduling and proposes a software project planning model that addresses the same. The model is described as follows.

A. Description of Employees

Software development is considered as vital human-intensive activity. A data storage such as an employee database to store employees’ information of wages, skills, and working constraints is required to manage employees. For efficient handling of tasks, it is required to assign employees to suitable tasks which involves human effort.

Suppose m employees are involved in the project, for the ith employee (i = 1, 2, …, m) the following attributes are considered:

- \(bs_i\) - Basic salary for an employee per time period (e.g., month)
- \(hs_i\) - Salary for an employee’s per-hour normal work
- \(ohs_i\) - Salary for an employee’s per-hour overtime work
- \(nh\) - Legal normal working hours per month
- \(maxh_i\) - Maximum working hours per month of the employee for a project
- \(\{join_i, leave_i\}\) - Time window when the employee is available for a project
- \(\{s_1, s_2, …, s_\phi\}\) - Skill set for an employee where \(\phi\) is the number of skills and \(s_j^i \in [0, 5]\) is the proficiency score of the jth skill. The skills can be documenting, C++ programming, GUI design, and any other technical abilities. \(s_j^i = 0\) means the employee does not possess the skill and \(s_j^i = 5\) means the employee is extremely good on that particular skill.

\[
\begin{align*}
\text{salary}_i &= bs_i + hours_i \cdot hs_i, \quad hours_i \leq nh, \\
&= bs_i + nh \cdot (hours_i - nh), \quad oh\text{s}_i, \quad nh < hours_i \leq maxh_i, \\
&= \infty, \quad hours_i > maxh_i.
\end{align*}
\]

B. Description of Tasks

A commonly used technique for task description is the task precedence graph (TPG). For a task \(t_j (j = 1, 2, \ldots, n)\), the following attributes should be considered:

- \(pm_j\) - The estimated work effort required for the task in person months. Several famous methods, for example, the COCOMO models [7], [8], can be adopted for work effort estimation.
- \(SK_j\) - The set of skills required by the task.
- \(maxhead_j\) - The maximum headcount for a task
- \(deadline_j\) and \(penalty_j\) - The deadline and penalty of the task.

Based on the above definitions, suppose \(wh_{ij}\) is the number of working hours of the ith employee for \(t_j\) at the \(t^{th}\) month; according to [6], [8], the achievement \(A_{ij}\) yielded by the employees for \(t_j\) at time \(t\) can be evaluated by the following steps:

1. The proficiency \(prof_{ij}\) of the ith employee for \(t_j\) can be evaluated by

\[
prof_{ij} = \prod s_j^i / 5 \quad (2)
\]

2. The total fitness \(F_j\) of the employees for \(t_j\) on the \(t^{th}\) month is given by

\[
F_j = \sum_{i=1}^{m} prof_{ij} \cdot wh_{ij} / \sum_{j=1}^{m} wh_{ij} \quad (3)
\]

3. Convert \(F_j\) to a cost driver value \(V = 8 - \text{round}(F_j \cdot 7 + 0.5)\), where the value of \(V\) belongs to 1-7. \(V = 1\) means the employees are the most suitable for the task and vice versa.

4. The achievement \(A_{ij}\) for \(t_j\) on the \(t^{th}\) month is calculated by

\[
A_{ij} = \sum_{i=1}^{m} wh_{ij} / V \quad (4)
\]

C. Planning Objective

As this software project planning involves task scheduling and employee allocation as the basic criteria, the plan has to determine the start time \(start_t\) and the finish time \(finish_t\) of each task \(t_i (i \in \{1, 2, \ldots, n\})\), and the working hours \(wh_{ij}\) of all employees where \(i \in \{1, 2, \ldots, m\}\) to the task \(t_j\) during the time window \(t \in [start_t, finish_t]\). The plan must satisfy the following constraints:
1. The processing order of tasks must definitely obey the precedence constraint defined by the TPG.

2. The working hours of the \(i^{th}\) employee per month must not exceed the limit \(\text{maxh}_i\), i.e.,

\[
\sum_{j=1}^{n} h_{ij} = \text{hours}_i \leq \text{maxh}_i, i = 1, 2, \ldots, \quad (5)
\]

3. The number of employees assigned to a task \(t_j\) is limited by the maximum headcount, i.e.,

\[
\sum_{i=1}^{m} \text{sign} \left( \sum_{w}^{w_{\text{start}_j}}^{w_{\text{finish}_j}} h_{ij} \right) \leq \text{maxhead}_j, \quad \text{where sign}(x) = 1; \text{if } x > 0, = 0; \text{if } x \leq 0. \]

4. All tasks have to be complete. In other words, for a task \(t_j\), the sum of the achievements for \(t_j\) during the time window \([\text{start}_j, \text{finish}_j]\) must satisfy

\[
\sum_{w}^{w_{\text{start}_j}}^{w_{\text{finish}_j}} A_{ij} \geq \text{pm}_j
\]

This paper considers cost minimization as the objective function, which is given by

\[
\min f = \sum_{i=1}^{n \text{end}} \text{salary}_i + \sum_{j=1}^{n \text{penalty}} j
\]

Where the first item denotes the salary expenditure and the second item denotes the total penalty.

III. RELATED WORK REVIEW

This paper resolve to propose a representation scheme with a jumper based ant colony optimization for project scheduling and staffing with an event-based scheduler approach for the above mentioned problem. In order to better depict why the representation scheme with jumper concept and the EBS are designed, we briefly review the existing software project planning models in this area.

A. The Resource Constrained Project Scheduling Problem Model

The RCPSP is a classical model in project management which is NP-hard. This involves scheduling the tasks of a project subject to precedence and resource constraints.

But, the RCPSP is still deficient for modeling the software project planning problem.

B. Employee Allocation Models

Employee allocation issues have drawn a considerable amount of research effort made in recent years and various models like nurse rostering and personnel planning have already been proposed. The employee allocation models for software project planning have to implicitly consider that a task can be conducted by an infinite number of employees and an employee can be assigned to an limitless number of tasks at one time, which is usually not the case in practice presently.

C. Multi Skill Scheduling Models

Considering the truth that software development involves a bunch of employees with many different skills.

After comparing with the models above mentioned, as the multiskill scheduling model is derived from the original RCPSP and couples the modeling of employees with various other skills, it also provides a more practical way for software project planning as it could deal with both the problems of task scheduling and employee allocation. The following situation may reduce the resource efficiency of the software project.

To model the task scheduling and human resource allocation problem in a more flexible way this model is used.

IV. THE EVENT-BASED SCHEDULER AND ITS REPRESENTATION

The drawbacks of mentioned review documents is met by the use of Event based scheduler. Like multischeduling problem in survey paper, we combine the representation of the list of tasks and employee allocation matrix so that the human resource allocation and project planning can be accomplished.

The algorithm is designed as
Initialize the no of human resources available during the whole scheduling window;
Set the starting time and the time when employees join or leave the software project as events;
T=1;
while the software project is not completed
if T is an event
find the tasks that can be constructed at time T and arrange these tasks into a sequence based on the order defined by the task list;
while the sequence is not empty;
set task as the first task in the sequence and remove that task from sequence;
for i=1 to noOfEmployees    //for each and every employee
if the planned working hours plannedWorkHours is not larger than the remainingAvailableWorkHours of the i-th employee at f;
remainingAvailableWorkHours^T = plannedWorkHours ;
else
remainingAvailableWorkHours^T is set to the remaining available working hours of the i-th employee at T;
end if
end for
end while
Else
the workload assignments are the same as those at T-1;
end if
end else
else
evaluate the final situation of the tasks at time T;
if there are any tasks completed at time T
set the time  T+1 as an event;
end if
T=T+1;
end while
end procedure

V. ANTS COLONY OPTIMIZATION APPROACH
A. Overview of the ACO Algorithm
The software project planning problem is solved using the ant colony optimisation approach in this paper. The actual idea of ACO algorithm is to simulate the behaviour of ants when there is a search for food by ants, they deposit some type of chemical on the path called pheromone and pass along with that. This serves as a medium of communication between ants. The ants sense the pheromone concentration and finds the path in search of food. This is one of the swarm intelligence phenomenon inspired by Dorigo, Developer of ACO.

It is applied to different optimisation problems and successfully proved its application. It is a work of dispatching finite group of ants to bring out solutions for a problem constructively. Fundamentally, an ACO algorithm is viewed as a medium and there is a repeated execution of three procedures:
A. Solution construction
B. Pheromone management
C. Daemon actions
By the above mentioned procedures and ACS variants, a 1. Flow Chart has been defined in this paper.
In the beginning, all the parameters are initialised. The build plans are set out then and the iteration being with jumper can find a best solution forwarding to any other iterations it needs. A plan for the problem includes task list and employee allocation matrix. Thus the solution construction procedure in the algorithm must include this with the training model and work experience of an employee.

Pheromone values are updated by by the global and the local updating rules. At last, local mutation procedure is designed to show the local search in improving the performance of algorithm.

B. Construction Procedure for Solutions

A feasible task list is built by an ant by maintaining an eligible set of tasks which satisfies the precedence constraints. The following steps are involved in the construction of task list are:

1. The tasks that should be implemented as the beginning of the project (i.e., the tasks that do not have precedence tasks) into eligibleSet.
2. For k (¼) 1 to n, process the following substeps
   - A task is selected from the eligibleSet and assign it to the kth position of the task list. In the selection rule, first a random number q uniformly distributed in [0, 1] is generated and it is compared with a parameter q_t.

   It is the selection rule designed based on the pseudo proportional rule in ACS. So the characteristic means that the ants have probability q_t.

   An example task list construction on TPG is shown.

   The construction of the employee allocation matrix is based on the following steps:
   - Step a: Set all values in the employee allocation matrix to 0.
   - Step b: For each task t_j, assign the workloads for t_j by the following substeps:
     1. Evaluate the value for all employees, where β is a parameter.
     2. Select an employee u from the set of employees which have not yet been chosen for t_j using the pseudorandom proportional rule in the ACS
     3. Assign the working hours for the employee selected to work for t_j using the pseudorandom proportional rule.
     4. If the number of the employees selected to work for t_j reaches maxhead_j, the workload assignment process for t_j is finished. Otherwise, go to Step b-2 to select other employees to work for t_j. After the workloads for all tasks have been assigned, the procedure for building the employee allocation matrix is complete.

VI. CONCLUSION

A new concept is developed for understanding and resolving the problem of human resource allocation and scheduling. Consideration of training model and experience of employees adds flexibility and the paper promises that this method is effective to be used for other applications also.
REFERENCES


