An Application of Ant Colony Optimization for Software Project Scheduling with Algorithm In Artificial Intelligence

1Ms.Minal C.Toley, 2 Prof.V.B.Bhagat

1M.E.First Year CSE
P.R.Pote COET, Amravati, Maharashtra, INDIA.

2Asst.Prof.CSE Department
P.R.Patil COET,Amravati,Maharashtra,INDIA.

Abstract

Ant colony optimization (ACO) is a population-based metaheuristic that can be used to find approximate solutions to difficult optimization problems. In ACO, a set of software agents called artificial ants search for good solutions to a given optimization problem research into developing effective computer aided techniques for planning software projects is important and challenging for software engineering. Different from projects in other fields, software projects are people-intensive activities and their related resources are mainly human resources. Thus, an adequate model for software project planning has to deal with not only the problem of project task scheduling but also the problem of human resource allocation. But as both of these two problems are difficult. The basic idea of the EBS is to adjust the allocation of employees at events and keep the allocation unchanged at non-events. With this strategy, the proposed method enables the modelling of resource conflict and task pre-emption and preserves preserves the flexibility in human resource allocation. To solve the planning problem, an ACO algorithm is further is used.

Keywords: Ant Colony Optimization (ACO), Software Project Scheduling, Event scheduler, Resource Allocation.

1.INTRODUCTION

With the rapid development of the software industry, software companies are now facing a highly competitive market. To succeed, companies have to make efficient project plans to reduce the cost of software construction [1].However, in medium to large-scale projects, the problem of project planning is very complex and challenging [2]. To plan a software project, the project manager needs to estimate the project workload and cost and decide the project schedule and resource allocation. For scheduling and staffing management, similarly to other projects (e.g., construction projects), management is usually conducted by project management tools and techniques. For example, traditional project management techniques like the program evaluation and review technique (PERT), the critical path method (CPM) [9], and the resource-constrained project scheduling problem (RCPSP) model [10] have been applied in software project planning. Although these methods are important and helpful, they are increasingly considered to be inadequate for modeling the unique characteristics of today’s software projects. The main reason is that, differently from other projects, a software project is a people-in intensive activity and its related resources are mainly human resources [4]. Different software project tasks require employees with different skills, and skill proficiency of employees significantly influences the efficiency of project execution. As such, 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.

2. Overview of the ACO Algorithm

To solve the software project planning problem, this paper proposes an ACO approach. The underlying idea of ACO is to simulate the foraging behaviour of ants. When ants search for food, they usually deposit a special chemical on the path they travel through. This kind of chemical, which is called pheromone, serves as a medium for ants to communicate with each other. By sensing the concentration of pheromone; other ants can follow the path to find the food. Inspired by this swarm intelligence phenomenon, ACO was developed by Dorigo et al. [7] and has been successfully applied to various optimization problems.

3. Solution construction

During each iteration of the algorithm, a group of ants set out to build solutions to the problem. Each ant builds a solution in a constructive manner by selecting components step by step to form a complete solution. The selections are made according to pheromone and heuristic information. In ACO, pheromone is a record of the past search experience of ants for guiding the following ants to make decisions. The components belonging to the best solutions found by the previous ants usually accumulate more pheromone, attracting more ants to select in future iterations. Heuristic is some problem-
dependent information that helps ants to have higher probabilities to select promising components in the solution construction procedure.

4. Flowchart

![Flowchart of ACO](image)

Fig 1: Flowchart of ACO.

5. Construction of Task List

A task list is an order of tasks (tp1; tp2; . . . ; tpn) that satisfies the precedence constraints defined by the TPG. Here we first define the pheromone and the heuristic for task list construction. Then the procedure for an ant to build a task list will be described in detail.

5.1 Construction procedure

To build a feasible task list, each ant maintains an eligible Set of the tasks that satisfy the precedence constraint. The construction includes the following steps:

Step a: Put the tasks that can be implemented at the beginning of the project (i.e., the tasks that do not have any precedence tasks) into the eligible Set.

Step b: For k = 1 to n, process the following sub steps b-1 and b-2, repeatedly:

Step b-1: Select a task from eligible Set and put the task to the kth position of the task list. In the selection rule, at first a random number q uniformly distributed in [0, 1] is generated and is compared with a parameter qt. If q < qt, then the task tj from the eligible Set with the largest value of \( \sum_{l=1}^{T} t(j,l) \cdot n(t) \) is chosen to put to the kth position. Otherwise, the task is selected using the roulette wheel selection scheme. The probability Pr(j, k) of selecting the task tj to the kth position in the roulette wheel selection is given.

5.2 Pheromone management

Along with the solution construction procedure, pheromone values are updated according to the performance of the solutions built by ants. Ants tend to deposit more pheromone to the components of better-performed solutions. At the beginning, all pheromone values are set to an initial value initial, where

\[
T\text{initial} = \frac{1}{\sum_{i=1}^{T}[b_{si}+\text{maxSallary}_i](\text{deadline})}
\]
Here maxsalary is the maximum possible salary of the ith employee per time period, which can be calculate
Maxsalary=
= [bsi+maxhihsi, 0<maxhi, nh
= Bsi+nh,hsi+(maxhi-nh)ohsi maxhi>nh
The denominator in [5] is actually an upper bound estimation of the objective function value. The upper bound is
estimated by assuming all employees contribute all of their working hours to the project during every time period and the
project is delayed by n time periods (months) with respect to a predefined deadline. Here, n is a rough worst-case
estimation that assumes each task leads to a one-month delay for the project. During the construction of the task list and
employee allocation matrix, immediately after a selection is made the local pheromone updating rule is applied to reduce the
corresponding pheromone values so that the following ants can have better chances of selecting other possible choices.
In this way, the algorithm manages to maintain diversity. Suppose the task tj is selected to the kth position of the task list;
the corresponding pheromone is updated by
Tt (j, k) = (1-p).Tt (j, k) +p.Tinitil
Where p is a parameter. Similarly, if the ith employee is assigned to the jth task with k working hours, the corresponding
pheromone is updated by
Tt (i, k) = (1-p).Tt (i, k) +p.Tinitil
Tt (i, j, k) = (1-p).Tt (i, j, k) +potential
At the end of each iteration, additional pheromone is deposited to the components associated with the best-so-far plans
found by the algorithm. All the pheromone values associated with the best-so-far plan. With the evolution of the
algorithm, the cost of the best-so-far plan found by the algorithm becomes lower and lower. Thus, the value of becomes
larger. In other words, more pheromone will be added to the components of the best so far plan to make them more
attractive for the ants.

5.3 Daemon action
Daemon actions mean the centralized operations that cannot be done by single ants.

6. Description of Tasks
In a software project, tasks can be any activity involved in software construction, for example, class design, programming,
and testing. A commonly used technique for task description is the task precedence graph (TPG) [6], [5]. A TPG is an
cyclic directed graph G(O; A). The set of nodes T= t1; t2; . . . ; tn corresponds to the set of tasks, where n is the number
of tasks in the project. The set of arcs represents the precedence relations among tasks. Anarcðti; tjÞ means ti is a direct
predecessor task of tj. Under the precedence constraint defined by the TPG, a task can only start when all of its direct
predecessor tasks have finished. The problem of task scheduling is to construct an efficient timetable for the implementation of tasks subject to the precedence constraints. For a task tj, the following attributes are considered pmj—The estimated work effort of the task in person months. Several famous methods, for example, the
COCOMO models [7], [8], can be adopted for work effort estimation.
. SKj—the set of skills required by the task.
. maxheadj—the maximum headcount for the task. In application, as too many employees working on the same task will
incur higher communication overhead and result in low efficiency, it is necessary to limit the number of employees for a task.

7. Description of Employees
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 ith employee (i¼1; 2; . . . ; m) the following attributes are considered.
. bsi—the basic salary for the employee per time period (e.g., month).
. hsi—the salary for the employee’s per-hour normal work.
. ohs i—the salary for the employee’s per-hour overtime works.
. nh—Legal normal working hours per month.
. maxhi—Maximum possible working hours per month of the employee for the project.
. sji— the time window when the employee is available for the project. fsii; s2i; . . . ; s the skill list for the employee.
. Pj—the proficiency score of the jth skill. Here the skills can be documenting, C++ programming, GUI design, and any
other technical abilities. Sj i ¼ 0 means the employee does not have the skill and sji ¼ 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. Let us suppose that the ith employee
devotes hoursti hours to the project at the tth month (hoursti\_maxhi). If hoursti is larger than the legal normal working hours nh, it implies that the employee works overtime for the project.

8. Representation and the Event-based Scheduler

To overcome the deficiencies of the above-mentioned models, this paper proposes a representation scheme with a novel event-based scheduler. Similarly to the representation in Yannibe and Amandi’s recent work [11] for the multiskill scheduling problem, we combine the task list representation [9] and the employee allocation matrix representation [10] so that both the problems of task scheduling and human resource allocation are addressed. The representation scheme is given by [12] Task list: tp1; tp2; . . . ; tpn.

However, differently from Yannibe and Amandi’s scheme [7] that directly uses the employee allocation matrix (10), the proposed representation scheme in (12) uses the planned employee allocation matrix. More specifically, the whij in (10) is the actual working hours of the ith employee for tj. But the pwhij in (12) is only the originally planned working hours of the ith employee for tj. Because the multiskill scheduling model has various restrictions and reduces the flexibility of human resource allocation, to relax such restrictions and enable task pre-emption we no longer fix the working hours of the ith employee for tj during the processing course of tj as pwhij. Instead, the actual working hours wht ij during the time t 2/startj; finish should be adjusted according to the urgency of tasks and the amount of available resources. The EBS is characterized by making new assignments at events. We regard the time t as an event if t satisfies any one of the following three conditions: 1) t ¼ 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.

Conclusion

The main motto of the above description is first, the method takes advantage of ACO to solve the complicated planning problem, the second one method introduces an event-based scheduler. Experimental results show that the representation scheme with the EBS is effective, and the proposed algorithm manages to yield better plans with lower costs and more stable workload assignments compared with other existing approaches. A new method for solving the software project planning problem has been developed.

Future Scope

In future research, it will be interesting to consider employee experience and the training model [6] to make the considered problem more comprehensive. Including uncertainty treatment in the software project planning model is also a promising research topic. In addition, since the First, model proposed in this paper 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.

Acknowledgement

The authors would like to acknowledge the editor and anonymous reviewers for their critical comments, which greatly improved this paper.

References

Author

Miss. Minal C. Toley is a scholar of ME, (Computer Science Engineering), at P. R. Pote COET, Amravati, under SGBAU, India.

Prof. Vaishali B. Bhagat is Asst Professor in P. R. Patil College of Engg. She did her M.E at Prof. Ram Meghe Institute of Technology & Research, Badnera.