International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.6 (2015), pp.25-38
http://dx.doi.org/10.14257/ijmue.2015.10.6.04

An Optimized Resolution for Software Project Planning with Improved Max–Min Ant System Algorithm

WanJiang Han[1], HeYang Jiang[2], TianBo Lu[1], XiaoYan Zhang[1] and WeiJian Li[2]

[1]School of Software Engineering, Beijing University of Posts and Telecommunication, Beijing 100876, China
[2]International School, Beijing University of Post and Telecommunication Beijing 100876, China
hanwanjiang@bupt.edu.cn

Abstract

Software Project Management (SPM) is one of the primary factors to software success or failure. SPM has been the bottleneck in software engineering area. We apply an improved Max–Min Ant System algorithm to Software Project Planning to make the appropriate worker-task assignment in a software project so the cost and duration of the project are minimized. Experimental results shows that using this Improved Ant Colony Optimization algorithm can obtain a feasible solution which can help us to get the appropriate PERT Graph and Gantt Chart of the software project, then we can obtain the minimized project duration and cost. So Software project management can been improved.

Keywords: *Ant Colony Optimization, Software project management, Software Project Planning, Max–Min Ant System Algorithm*

1. Introduction

Software project management is a complex and difficulty job, especially in planning and controlling. Software project planning can be one of the most critical activities in the SPM process. Without a realistic and objective software project plan, the software development process cannot be managed in an effective way. So, we need to optimize project planning. The Software Project Scheduling Problem (SPSP) is an specific Project Scheduling Problem [1-2], which makes the appropriate employee-job assignment that minimizes cost and duration for the whole project, and the task-precedence and resource constraints are satisfied [3-5]. While in Planning Software Project, we should assign appropriate employees to project tasks according to their skills. The optimal solution can minimize the duration and cost of the project.

There are three general software project planning approaches: past experience, standard guidelines, and support tools. Experienced managers rely upon their past experiences and experience of successful managers to create plans. Often past finished project documents and guidelines are available to use as models for project plans.

The ant colony optimization (ACO) heuristic presented by Colorni *et al.* [6], and later extended by several studies [7], uses as warm intelligence approach to solve the traveling salesman problem. In this well-known NP-Hard problem, a set of n cities must be visited by one single traveling salesman. The goal is to minimize the total distance length.[6,7], and Gambardella and Dorigo [8] and other shave created computational agents that mimic the behavior of real ants going from their nests to a food source. To do this, the ‘‘artificial ants’’ are placed on a graph and forced to move. Each single movement of the ant on the graph generates another piece of the solution. The set of moves generates the full problem solution [9-10] .

ISSN: 1975-0080 IJMUE

Project scheduling problem, which has been widely studied since early 1960's, is to determine the schedule of allocating resources so as to balance the total cost and the completion time. The general resource-constrained project scheduling problem (RCPSP) arises when a set of interrelated activities (precedence relations) is given and when each activity can be performed in one of the several ways [11].

Alba firstly presented the model of the SPSP and Genetic Algorithms (GA) [12]. Chang proposed a Time-line based model for SPSP using Genetic Algorithms [13]. Xiao proposed the Ant Colony System (ACS) to solve SPSP [14], and later Crawford gave a new solution by using Max− Min Ant Systems (MMAS) in Crawford, Soto, Johnson, and Monfroy [15]. Luna makes a scalability analysis of multi-objective metaheuristics solving SPSP [16].

Ant Colony Optimization algorithm (ACO) algorithms make use of simple agents called ants which iteratively construct candidate solutions to a combinatorial optimization problem. The ants' solution construction is guided by pheromone trails and problem-dependent heuristic information [17].

The Max–Min Ant System (MMAS) is an ACO algorithm that builds directly upon Ant System, incorporating a much more aggressive pheromone update procedure and mechanisms to avoid search stagnation.

MMAS establishes a minimum and maximum value for the pheromone, and provides that only the best ant can update the pheromone trail. MMAS can achieve better search performance than ACO algorithm by increasing the exploitation of the best solution found during the search process [17].

This paper deals with software project planning, especially scheduling problem. Based on the work of Broderick Crawford [18], R.F. Tavares Neto [19], we presents an optimized resolution which consists in improving Max− Min Ant System algorithm. Using this algorithm to solve the Software Project Scheduling Problem, conducting experiments to show how to build project Critical Path Method (CPM) graph, Gantt Chart and Cost baseline.

2. The Model of Software Project Planning

The RCPSP is to solve NP- hard problem. It is used to schedule the tasks of a project subject to precedence and resource constraints. To generate a schedule, it needs to determine order of tasks which satisfy task precedence constraint and make task list [20].

Software project planning includes developing a worker-task schedule for a software project. We should consider employee skills and remunerations and assign them to project tasks according to task requirements [3-4, 14, 18].

Our goal is to obtain the result of minimized cost and duration for the whole project

In the model of software project planning, there are three important resources: tasks, employees, and skills. The tasks are the job needed to complete the project. The employees are the workers who have some skills and work in the task which requires a set of skills. This model can make an appropriate allocation by the skills required for the tasks and the skills of employees.

Assume P is a project, which has a set of tasks, represented by $T = \{t_1, . . . , t_n\}$, where n is the total number of tasks. This project needs a set of employees to work on the project tasks. The set of employees is defined as $E = \{e_1, e_2, . . . , e_m\}$, where m is the total number of Employee, Each task requires a set of skills and an effort. We present a set of skills as $S = \{s_1, s_2, . . . , s_k\}$ where k is the total number of skills.

Each task has two attributes: t_j^{sk} and t_j^{eff} . t_j^{sk} is a set of skills for the task j, t_j^{eff} represents the workload of the task j.

Now we define M as solution matrix, which is employee allocation model, shown as Eq. (1).

$$\begin{vmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ m_{m1} & m_{m2} & \cdots & m_{mn} \end{vmatrix} \tag{1}$$

Where $m_{ij} \in [0,1]$, which represents the degree of dedication of employee i to task j. If $m_{ij} = 0.25$, the employee it takes 25 percent of his time on the project.

It is a $n \times m$ matrix of Task and Employee where employees are assigned to each Task, which size is the dimension of matrix corresponded to the employee number and the task number.

The solutions generated in this matrix for the project will map all employees to the project task and then generates an optimal permutation of software tasks, in the end we can get the best optimal solution with minimum cost and duration for the whole project and we can plan the software project better.

To generate an optimized solution, corresponding constraints should be considered. Firstly, the Matrix described above is feasible, secondly, the cost and duration for the whole project is minimal. Thirdly, project risk is the lowest.

In order to obtain a feasible matrix M, we define some constraints. For it, one employee or more are assigned to all project tasks, presented in Eq. (2).

$$\begin{vmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ m_{m1} & m_{m2} & \cdots & m_{mn} \end{vmatrix} > 0 \tag{2}$$

Other constraint is that the employee assigned to the task has all the necessary skills to complete the task. This is presented in Eq. (3)

$$t_j^{sk} \subseteq \cup e_i^{sk} \tag{3}$$

Where $j \in \{1, \ldots, n\}$, n is the task number.

So, Eq. (4) describes a feasible matrix M, which shows dedication of employees to very task. It can give us a feasible solution to plan a project.

$$\begin{vmatrix} 0.5 & 0 & 0.25 & 1.0 & 0.5 \\ 1.0 & 1.0 & 1 & 0 & 1.0 \end{vmatrix} \tag{4}$$

3. The Fitness Function

To generate an optimized solution, we will present a fitness function to appraise the quality of the solution. We will take the following steps [18].

Step 1: Building task-precedence-graph

Figure.1 shows the task precedence graph (TPG) of a project and Table 1 describes their necessary skills and effort.

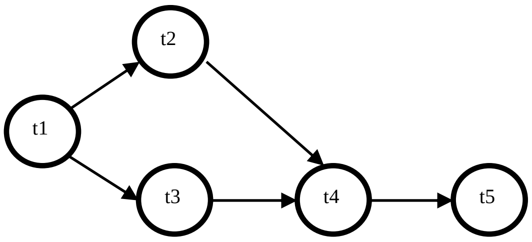


Figure 1. TGP with $T = \{t_1, t_2, t_3, t_4, t_5\}$

Table 1. Necessary Skills and Effort for Tasks

	t_1	t_2	t_3	t_4	t_5
Skills (t^{sk})	$\{s_1, s_2\}$	$\{s_2\}$	$\{s_1, s_3\}$	$\{s_1, s_2, s_3\}$	$\{s_1\}$
Effort (t^{eff})	8	18	23	15	9

Step 2: Employee allocation

For this project, we have a set of employees $E = \{e_1, e_2\}$, and each one has a set of skills, maximum degree of dedication $e^{\max d}$, and remuneration e^{rem}, illustrated in Table 2.

Table 2. Skills, Dedication, and Remuneration for Employees

	e_1	e_2
Skills (e^{sk})	$\{s_1, s_2, s_3\}$	$\{s_1, s_2\}$
maximum degree of dedication($e^{\max d}$)	0.5	1.0
Remuneration(e^{rem})	\$5000	\$6000

Step 3: The duration of all tasks

Assume the duration of task is, where, n is the task number, which can be calculated according to the Eq. (5).

$$t_j^{len} = \frac{t_j^{eff}}{\sum_{i=1}^{m} m_{ij}} \tag{5}$$

here m is the employee number.

For task j, the initialization time is presented as t_j^{init} and the termination time is presented as t_j^{term}. we can use the precedence relationships described as TPG to calculate t_j^{init} and t_j^{term}, as in Eq. (6) and Eq. (7).

In Eq. (6), if a task is without precedence, the initialization time is $t_j^{init} = 0$. If tasks with precedence, the termination time for all previous tasks must be calculated firstly.

$$t_j^{init} = \begin{cases} 0 & if - No - \text{precedence} \\ \max\{t_l^{term} | (t_l, t_j) \in E\} & else \end{cases} \tag{6}$$

$$t_j^{\text{term}} = t_j^{init} + t_j^{len} \tag{7}$$

Using the initialization time, the termination time and the duration for task j, we can generate a Critical Path Method (CPM) graph and Gantt chart. To calculate the total length of the project p^{len}, we only need termination time of last task, this is described in Eq. (8).

$$p^{len} = \max\{t_k^{term}, \forall k \neq j, \{t_j, t_k\}\} \tag{8}$$

Step 4: The cost of the project

To calculate the cost of the whole project, we need firstly to calculate the cost of each task as $t_j^{\cos}$ using Eq. (9), then calculate the total cost $p^{\cos}$ according to the Eq. (10).

$$t_j^{\cos} = \sum_{i=1}^{m} e_i^{rem} m_{ij} t_j^{len} \tag{9}$$

Where m is the total number of Employee.

$$p^{\cos} = \sum_{j=1}^{n} t_j^{\cos} \tag{10}$$

Where n is the total number of tasks.

Step 5: The fitness function

Our goal is to minimize the total duration and the total cost, so a fitness function is proposed, as in Eq. (11), where $w^{\cos}$ and w^{len} are the importance of $p^{\cos}$ and p^{len}, which meet the different magnitudes. For this, $(\text{Average cost})^{-1}$ can be the initialization of $w^{\cos}$, and $(\text{Average length})^{-1}$ can be the initialization of p^{len}. Then multiplying the parameters by the corresponding $p^{\cos}$ and p^{len}, the units are canceled and the values are in the same magnitude.

$$f(x) = w^{\cos} p^{\cos} + w^{len} p^{len} \tag{11}$$

Step 6: Risk element

An element not considered is project risk that may increase the cost and duration of the project. We can define project risk as t_i^{risk}. To calculate it, we use a function risk exposure, as is presented in Eq. (12).

$$t_i^{risk} = p_i^{risk} \times I_i^{risk} \tag{12}$$

Where p_i^{risk} is probability of risk occurrence I_i^{risk} is risk impact on the task.

Now we can calculate the project risk in the whole project using the Eq. (13).

$$P^{risk} = \sum_{i=0}^{n} t_i^{risk} \tag{13}$$

We can determine if the solution is feasible. In this case, is feasible when the solution can complete all tasks, and the fitness function is minimized, and no risk or least risk, that means the is minimized.

4. An Improved MMSA Algorithm for Software Project Plan

In principle, ACO algorithms can be applied to any combinatorial optimization problem by defining solution components which the ants use to iteratively construct candidate solutions and on which they may deposit pheromone[18-19].

The MMAS strategy has a well-defined pheromone range imposed by two constants that delimit the upper and lower bounds. Together with an elitist strategy, which allows only the ants with best solutions to deposit pheromones, this algorithm is, according to Dorigo and Blum, becoming a highly successful ACO variation[9][10].

This section describe how to adapt this improved Max-Min Ant System (IM-MMSA) Algorithm to software project plan. Firstly, establish an appropriate construction graph, then define the use of pheromone. This algorithm makes the association of employees to project tasks.

4.1. Construction Graph for Software Project

The construction graph is the graph that the ants travel through. Then a solution can be constructed. The ants start from an initial node and then select the nodes according to a probability function. This function is given by the pheromone and heuristic information of our problem, their relative influence is given by α and β , respectively [20-23].

We construct a directed graph by using the heuristic information and pheromone to. The employees should be assigned to a project task, and their dedication to a task is represented in the graph TGP. With this representation, we can define the construction graph. The proposed construction graph represents the association of employee and their dedication to a task. This representation is constructed for each task in the TGP and is divided into a graph with node and edge. Therefore, the construction graph includes all employees and their ratio of dedication contributions for the project task. So we define a variable “density” which is density of nodes, presented as following.

$$\text{density} = \frac{1}{\text{least}} + 1$$

Where” least” is the lowest degree of dedication to a task. This structure is presented in Figure. 2. The employees dedication to a task can be 0 or integer multiple of” least”. Firstly, generate a start node and insert into first column. Secondly, generate the i columns, with $i = \{1, . . . , m\}$. Each column consists in $density$ nodes. Third, construct an end node and put into the last column. Fourth, constructs edges, add edges between columns. The ants will travel from the start node to the end node choosing edges from the column 1 to column m with no returning. The ants choose only one node each column.

When a tour is finished, we can obtain the dedication of employee i to task j, presented in Eq. (13) which is the element $m(i, j)$ of solution Matrix.

$$m(i, j) = k \times least \tag{13}$$

Where k is the node j in column i

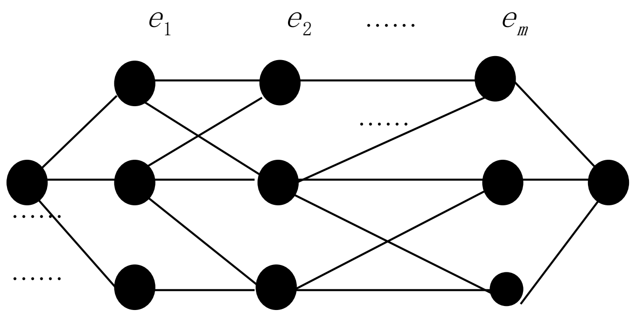


Figure 2. Construction Graph is a Matrix CG for a Task

The ants travel through the construction graph, based on a probabilistic decision to select ways. This probabilistic choice is biased by the pheromone trail τ_{ij} and by a locally available heuristic information η_{ij}. So, ant t choose employee i in column j with a probability, presented in Eq. (14).

$$p_{ij}^{t} = \begin{cases} \dfrac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{l=0}^{den} \tau_{ij}^{\alpha} \eta_{ij}^{\beta}} & if j \in \{1, \ldots, d\} \\ 0 & otherwise \end{cases} \tag{14}$$

where τ_{ij} is the pheromone and η_{ij} is the heuristic information of the problem on the path between node i to j in the graph CG for task t. α and β are two fixed parameters, which are used to determine the pheromone and heuristics influences.

4.2. The Pheromone Update Strategy

The pheromone update strategy is shown in Eq. (15) and (16), in which the updated value of the pheromone on the trail. This update strategy will take into account all solutions generated by the ants. And it enables the pheromone deposit only on trails used by ants that found the best solution [20][24].

$$\tau_{ij} = \rho\tau_{ij} + (1-\rho)\Delta\tau^{ipd} \tag{15}$$

$$\Delta\tau^{upd} = (w^{\cos} p^{\cos} + w^{len} p^{len} + w^{risk} p^{risk})^{-1} \tag{16}$$

where $p^{\cos}$ is the total cost of software project, p^{len} is the total duration, p^{risk} is project risk , $w^{\cos}$, w^{len} and w^{risk} are the importance of $p^{\cos}$, p^{len} *and* p^{risk}. ρ is a rate of evaporation $\rho \in [0,1]$. If ρ is high, the new pheromone value is less influenced by $\Delta\tau^{upd}$, but much influenced by the previous pheromone value, vice versa. And $\Delta\tau^{upd}$ it is associated with quality of the current solution of upd ant. upd ant is iteration-best ant.

We can use an updating pheromone strategy considering duration and cost of the project as (16).

4.3. Improved MMAS Algorithm

To improve MMAS Algorithm, a minimum pheromone threshold τ_{hold} value is used to monitor the search performance of ants [25-29]. The τ_{hold} value should not be higher than the maximum pheromone trail limit ($\tau_{\max}$) and lower than the $\tau_{\min}$ value, as shown in Eq. (17). The τ_{hold} value is set at a particular value during the start of the algorithm, and changed according to the search performance. Whenever τ_{ij} is lower than τ_{hold} , the value of that particular partial solution will be changed to $\tau_{\max}$ as shown in Eq.(18). Thereafter, This improved Max-Min Ant System Algorithm allows that particular partial solution to be involved in the successive solution construction steps.

$$\tau_{\min} < \tau_{\mathrm{hold}} < \tau_{\max} \tag{17}$$

$$\tau_{ij} = \tau_{\max} \text{ when } \tau_{ij} < \tau_{\mathrm{hold}} \tag{18}$$

The benefit of this improvement is that, whenever the τ_{ij} value drops to a low level, the particular partial solution will not be excluded from the successive solution constructs. This helps to increase the exploration or diversification of the search.

4.4. Algorithm Description

This Improved MMSA Algorithm for Software Project Plan is Described as Follows.

```
Input: The information of  Software project
 Initialize the pheromone values , $\tau_{\max}$, $\tau_{\mathrm{hold}}$ , $\tau_{\min}$
 While termination conditions not met do
   For g = 1 to G do
   initialize process  of ant
     For j=1 to m  do
       For task =1 to t do
     ant $Ant_j$ travel on the matrix CG choosing employee and their dedication to a task
$task_{task}$
       storing partial route for $Ant_j$
 End for
    ensure feasible Matrix t, using Eq. (2).
    calculate the fitness of the solution, using Eq. (11).
   End for  select the best solution
  update pheromone values as Eq. (15), (16),(17),(18).
  End for
 select the best global solution (optional)
 apply pheromone update for global-best
 End while (iterations is complete)
 Output: An optimized solution
```

Firstly, the algorithm reads the project information, which provides the necessary data of software project, such as tasks, required skills, employees, task precedence information, the set of skills of each employee, and their remunerations, also risk information. Then, we have to split operation to the task and then using above algorithm to generate solutions. To obtain the optimal solutions, we use Eq. (11) that is the fitness function, which minimizes the cost and duration for the whole project.

From the optimal solution, we can get the dedication matrix, then we can plan software project.

5. Experimental Results

In this section, we will show an experiment and its results. This experiment is to plan a software project where a set of employees must be assigned to a set of tasks, so all tasks are completed by employees who have the necessary skills to accomplish tasks. This assignment should minimize the cost and duration of the whole project. The employees have remuneration and several skills and they can work on multiple tasks during the workday.

First, we describe software project instance. The set of all skills associated with this software project is $\{s_1, s_2, s_3\}$, the set of tasks necessary for this project is $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. Figure. 3 shows the task- precedence-graph TPG. The set of employees is $\{e_1, e_2, e_3, e_4\}$, which means 4 employees working on this project, their necessary skills and remuneration are illustrated in Figure.4. That means a subset of the union of the skills the employees assigned to the task and their cost. The most relevant resource in project Scheduling is the project employees.

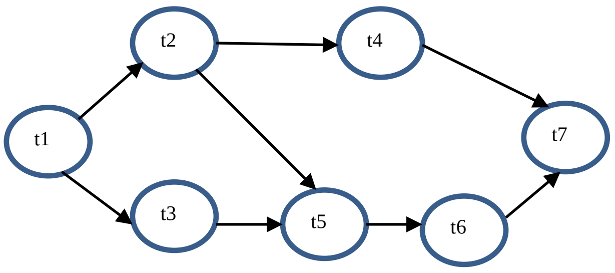


Figure 3. TGP with $T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$

For this project, These tasks $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ are “Planning”, “High-level design”, “Low-level design” , “ Testcase design”,“Coding” , “System Testing ”, “Acceptance Testing ”, respectively.

$$e_1^{sk} = \{s_1, s_2, s_3\} \; e_2^{sk} = \{s_1, s_2, s_3\}, e_3^{sk} = \{s_1, s_2\} \; e_4^{sk} = \{s_2, s_3\}$$
$$e_1^{rem} = \$6000, e_2^{rem} = \$6000, e_3^{rem} = \$5000, e_4^{rem} = \$5000$$

Figure 4. Skills and Remuneration of Employees

For these tasks in Figure. 3, we can see their necessary skills and effort in Figure. 5 and in Figure. 6.

$$t_1^{sk} = \{s_1, s_2, s_3\},\ t_2^{sk} = \{s_1, s_2\},\ t_3^{sk} = \{s_2, s_3\}, t_4^{sk} = \{s_2\},$$
$$t_5^{sk} = \{s_1, s_2, s_3\}, t_6^{sk} = \{s_1, s_2\}, t_7^{sk} = \{s_2\}$$

Figure 5. Necessary Skills for Tasks

$$t_1^{eff} = 4, t_2^{eff} = 6, t_3^{eff} = 8, t_4^{eff} = 6,$$
$$t_5^{eff} = 8,\ t_6^{eff} = 10, t_7^{eff} = 16$$

Figure 6. Necessary Efforts for Tasks

The algorithm was implemented in Java using a Intel core i8, 1.0 Ghz, 8 GB of RAM PC running Windows 8 Professional. And ran 5 trials for this instance and we report the average value. In the end, we will find feasible solutions.

5.1. Model Parameter

In our experiment, model parameters should follow the rule with higher availability or shorter execution time. To choose appropriate parameters, we conducted a series of experiments and came cross lots of references and literatures. We choose the parameters as follows: $\alpha = 1$, $\beta = 2$, $\rho = 0.02$, $m = 200$ and number of iterations is 1000.

5.2. Experimental Report

In this section, experimental results are presented. In Figure 7 shows the fitness with different iteration by 5 trials. We obtain the best solutions from iteration 400. From the best solutions, we can get the employee dedication to each task, as in (19).

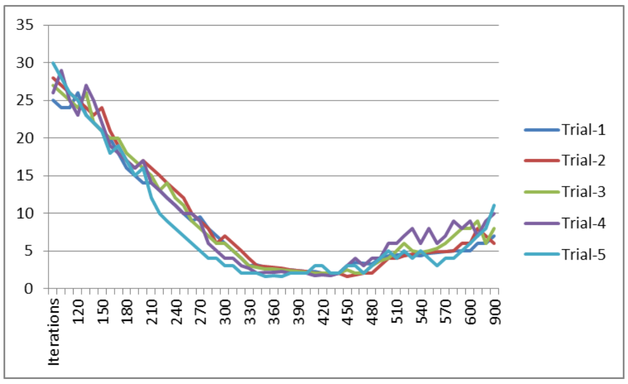


Figure 7. The Fitness Function

$$\begin{vmatrix} 1 & 0.5 & 0.5 & 0 & 1 & 1 & 1 \\ 1 & 0.5 & 0.5 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0.5 & 0 & 0.5 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{vmatrix} \qquad (19)$$

From (19) and task effort described above, We can compute the length time for each task as $t_j^{len}, j \in \{1,2,3,4,5,6,7\}$, presented in Figure 8. Then we can compute the cost of each task, presented in Figure. 9. In this way, we can obtain the initialization time and the termination time for each task, so we can get the total length of the project p^{len} =18 and the total cost $p^{\cos} = \$340000$.

$$\boxed{\begin{array}{l} t_1^{len} = 2,,,, t_2^{len} = 3, t_3^{len} = 4, t_4^{len} = 6 \\ t_5^{len} = 4 \ t_6^{len} = 4 \ t_7^{len} = 4 \end{array}}$$

Figure 8. The Duration for Each Task

$$\boxed{\begin{array}{l} t_1^{\cos} = \$24000,,,, t_2^{\cos} = \$33000, t_3^{\cos} = \$44000 \\ t_4^{\cos} = \$45000 , t_5^{\cos} = \$48000 , t_6^{\cos} = \$58000, t_7^{\cos} = \$88000 \end{array}}$$

Figure 9. The Duration for Each Task

From these results, we can develop the project CPM graph displayed in Figure 10 and Gantt chart showed in Figure 11. In the Figure 10, we can see the critical path is $t_1 -> t_2 -> t_5 -> t_6 -> t_7$, so the shortest amount of time needed to complete this project is $p^{len} = 18$.

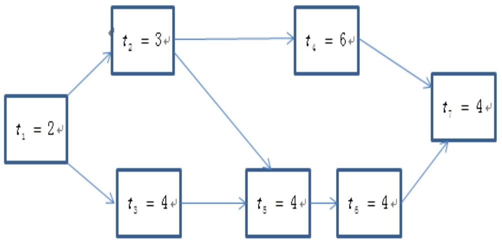
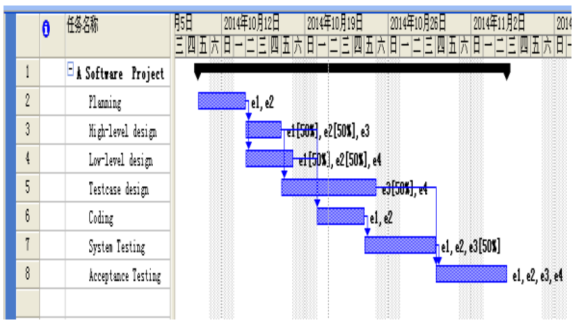


Figure 10. PERT Graph



	任务名称	
1	A Software Project	
2	Planning	e1, e2
3	High-level design	e1[50%], e2[50%], e3
4	Low-level design	e1[50%], e2[50%], e4
5	Testcase design	e3[50%], e4
6	Coding	e1, e2
7	System Testing	e1, e2, e3[50%]
8	Acceptance Testing	e1, e2, e3, e4

月5日
2014年10月12日
2014年10月19日
2014年10月26日
2014年11月2日
三 四 五 六 日 一 二 三 四 五 六 日 一 二 三 四 五 六 日 一 二 三 四 五 六 日 一 二 三 四 五 六 日

Figure 11. Gantt Chart

In order to control project better, we should obtain cost baseline. For this, according to the Gantt chart and the cost of the whole project, we can get the cost baseline as Figure 12.

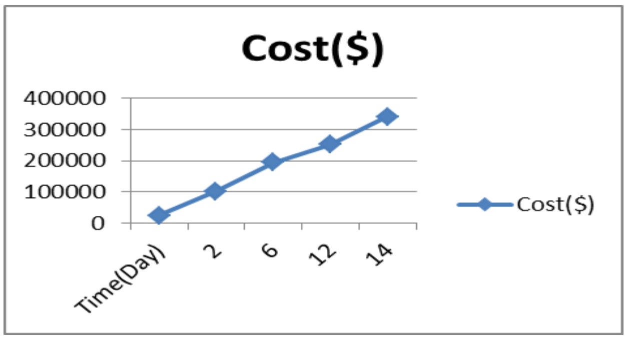


Figure 12. Cost Baseline

Now we have finished the project plan.

6. Conclusions

This paper presented an optimized resolution for Software Project Plan, using an improved Max–Min Ant System algorithm. By it we can obtain a feasible solution. Through this solution and TPG, we can compute the length time for each task, draw PERT chart, then get the critical path, get the total length of the project. Then we can arrange project schedule by tasks, their required skills, and employees. In the last, we can achieve an optimized solution. The experiments were performed and showed the results. As future work, we will constantly improve the algorithm and solution means, and will integrate means and project arrangement, so we can implement project arrangement autonomously.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (Grant No. 61170273).

References

[1] Laalaoui, Y. and Bouguila, N., "Pre-run-time scheduling in real-time systems:Current researches and artificial intelligence perspectives", Expert Systems with Applications, vol. 41 no. 5, **(2014)**, pp. 2196 – 2210.

[2] Chen, R.-M., "Particle swarm optimization with justification and designed mechanisms for resource-constrained project scheduling problem", Expert Systems with Applications, vol. 38 no. 6, **(2011)**, pp. 7102–7111.

[3] Alba, E. and Chicano, F., "Software project management with gas", Information Sciences, vol. 177 no. 1, **(2007), pp.** 2380 – 2401.

[4] C. K. Chang, Hsin Yi J., Yu D., Dan Z. and Yujia G., "Time-line based model for software project scheduling with genetic algorithms", Information and Software Technology, vol. 50 no. 11, **(2008)** pp. 1142 – 1154.

[5] Nan, N. and Harter, D. E., "Impact of budget and schedule pressure on software development cycle time and effort", IEEE Transactions on Software Engineering, vol. 35 no. 5, **(2009)**, pp. 624 – 637.

[6] Colorni A., Dorigo M. and Maniezzo V., "Distributed optimization by ant colonies", In: European conference of artificial life; **(1991)**, pp.134–142.

[7] Dorigo M., Maniezzo V. and Colorni A., "The ant system:optimization by acolony of cooperating agents", IEEE Transactionson Systems, Man,andCybernetics—Part B 1996; vol. 26, **(1996)**, pp. 29–41.

[8] Gambardella L. M. and Dorigo M., "Solving symmetric and asymmetric tsps. By ant colonies", In:Proceedings of the IEEE conference on evolutionary computation, ICEC96. IEEE Press ; **(1996)**, pp. 622–627.

[9] R. F. Tavares Neto and M. Godinho Filho, “An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed”, Computers & Operations Research, vol. 38, **(2011)**, pp. 1286–1293

[10] Dorigo M. and Blum C. “Ant colony optimization theory: a survey”, Theoretical Computer Science vol. 344 no. 2–3, **(2005)**, pp. 243–78.

[11] Siamak Baradaran , S. M .T. Fatemi Ghomi , Mahdi Mobini and S.S. Hashemin, “A hybrid scatter search approach for resource-constrained project scheduling problem in PERT-type networks”, Advances in Engineering Software, vol. 41, **(2010)**, pp. 966 – 975

[12] Alba E. and Chicano F., “Software project management with gas”, Information Sciences, vol. 177 no. 1, **(2007)**, pp. 2380 – 2401.

[13] Carl K. C., Hsin-yi J., Yu Di, Dan Zhu & Yujia Ge, “Time-line based model for software project scheduling with genetic algorithms”, Information and Software Technology, vol. 50 no. 11, **(2008)**, pp. 1142 – 1154.

[14] Xiao J., Ao X. T., and Tang Y., “Solving software project scheduling problems with ant colony optimization”, Computers and Operations Research, vol. 40 no. 1, **(2013)**, pp. 33 – 46.

[15] Crawford B., Soto R., Johnson F. and Monfroy E., “Ants can schedule software projects. In C. Stephanidis (Ed.). HCI international 2013 – posters’ extended abstracts, communications in computer and information science”, Berlin Heidelberg: Springer, vol. 373, **(2013)**, pp. 635 – 639

[16] Luna F., González-álvarez D. L., Chicano F., and Vega-Rodríguez M. A., “The software project scheduling problem: A scalability analysis of multi-objective metaheuristics”, Applied Soft Computing, vol. 15 no. 0, **(2014)**, pp. 136 – 148.

[17] Stützle T. and Hoos, H. H., “Max-min ant system”. Future Generation Computer Systems, vol. 16 no. 8, **(2000)**, pp. 889 – 914.

[18] Broderick C., Ricardo S., Franklin J., Eric M., Fernando P., “A Max – Min Ant System algorithm to solve the Software Project Scheduling Problem”, Expert Systems with Applications, vol. 41 **(2014)**, pp. 6634 – 6645

[19] R. F. Tavares Neto, M. Godinho Filho, “An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed”, Computers & Operations Research, vol. 38, **(2011)**, pp. 1286–1293.

[20] Kishor N. V. and Dinesh B. H., “Software Project Planning Using Ant Colony Optimization (SPP-ACO)”, International Journal of Computer Science And Technology, vol. 4, no. 4, **(2013)**.

[21] Berrichi A., Yalaoui F., Amodeo L. and Mezghiche M., “Bi-objective ant colony optimization approach to optimize production and maintenance scheduling”, Computers and Operations Research, vol. 37 no. 9, **(2010)**, pp. 1584 – 1596.

[22] Dorigo M. and Di Caro G., “Ant colony optimization: A new meta-heuristic. In Proceedings of the 1999 congress on evolutionary computation”, IEEE Press, **(1999),** pp. 1470 – 1477

[23] Dorigo M., Maniezzo V. and Colorni A., “Ant system: Optimization by a colony of cooperating agents”, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 26 no. 1, **(1996)**, pp. 29 – 41.

[24] Xun P., Xian P. Y., Xian L. L., “Optimal Strategies for Jobs Scheduling in Grid Using Max-Min Ant System”, Journal of Convergence Information Technology(JCIT), vol. 7 no. 5, **(2012)**.

[25] Phen C. S., Kuan Y. W., Komarudin, “A Technique for Improving the Max-Min Ant System Algorithm” ,Proceedings of the International Conference on Computer and Communication Engineering 2008 May 13-15, **(2008)**, Kuala Lumpur, Malaysia.

[26] A. R. S. Amaral, “On the exact solution of a facility layout problem”, European Journal of Operational Research, vol. 173 no. 2, **(2007)**, pp. 508-518.

[27] E. Duman and I. Or, “The quadratic assignment problem in the context of the printed circuit board assembly process”, Computers & Operations Research, vol. 34 no. 1, **(2007)**, pp. 163-179.

[28] Barreto A., Barros M. d. O., and Werner C. M. L., “Staffing a software project: A constraint satisfaction and optimization-based approach”, Computers and Operations Research, vol. 35 no. 10, **(2008)**, pp. 3073 – 3089.

[29] Ching-seh W., Wei-chun C., Ishwar K. S., “A Metric-Based Multi-Agent System for Software Project Management”, Eigth IEEE/ACIS International Conference on Computer and Information Science, **(2009).**

Authors

Wan-Jiang HAN, she was born in HeiLongJiang province, China, 1967. She received her Bachelor Degree in Computer Science from Hei Long Jiang University in 1989 and her Master Degree in Automation from Harbin Institute of Technology in 1992.

She is an assistant professor in School Of Software Engineering, Beijing University of Posts and Telecommunication, China. Her technical interests include software project management and software process improvement.

Tian-Bo Lu, he was born in Guizhou Province, China, 1977. He received his Master Degree in computer science from Wuhan University in 2003 and his PH.D Degree in computer science from the Institute of Computing Technology of the Chinese Academy of Sciences in 2006.

He is an Associate professor in School of Software Engineering, Beijing University of Posts and Telecommunications, China. His technical interests include information and network security, trusted software and P2P computing.

Xiao-Yan Zhang, she was born in Shandong Province, China, 1973. She received her Master Degree in Computer Application in 1997 and her PH.D Degree in Communication and information system from Beijing University of Posts and Telecommunication, China, in 2011.

She is an Associate professor in School of Software Engineering, Beijing University of Posts and Telecommunications, Beijing, China. Her technical interests include software cost estimation and software process improvement.