Applying a $MAX$-$MIN$ Ant System with a Dynamic Roulette Wheel Strategy to Software Release Planning

Yu-Qing Huang¹, Chuan-Wen Chiang ², Cheng-Hsu Huang³

¹Department of Computer Science & Information Engineering, National Central University, Taoyuan County 32001, Taiwan
²Department of Computer and Communication Engineering, National Kaohsiung First University of Science and Technology, Nanzih, Kaohsiung 811, Taiwan, R.O.C.
³Department of Computer Science and Information Engineering, Hwa Hsia Institute of Technology, Taiwan

u9115903@ccms.nkfust.edu.tw

Abstract - In this study, the software releasing planning (SRP) problem resulted from incremental software development is considered. Software releasing planning has been proven to be a NP-complete problem. Owing to the intractable nature of the problem, a heuristic approach based on ant colony optimization (ACO) can be applied to obtain satisfactory suboptimal solutions within a reasonable amount of computational cost. However, most realistic ACO-based approaches for software releasing planning still remain to be improved. A novel ant-inspired search algorithm is therefore proposed. The proposed algorithm, namely $MMAS_{DRW}$-$SRP$, adopts a dynamic roulette wheel strategy for giving a sophisticated balance between intensification and diversification, thereby improving the quality of solutions obtained. The performance of $MMAS_{DRW}$-$SRP$ is demonstrated by comparing it against conventional ACO-based approaches. Experimental results indicate that the proposed $MMAS_{DRW}$-$SRP$ algorithm performs significantly better than the competitive approaches.

Index Terms - software release planning (SRP), ant colony optimization (ACO), $MAX$-$MIN$ Ant System ($MMAS$).

1. Introduction

Software project management aims to plan and monitor software projects using management knowledge and skills while working within the constraints of time, resources and budgets imposed by project environment and stakeholders [1]. The incremental software development model is one of the popular product development models in recent years. Release planning addresses the assignment of features to a sequence of consecutive releases such that the related resources and budgets constraints are satisfied [2]. Since the software release planning has been proven to be a NP-complete problem [3]. There are many studies have focused on various metaheuristic algorithms to acquire near-optimal solutions within a reasonable amount of computation time. For example, the famous Ant System (AS) proposed by Marco Dorigo is inspired from the foraging behavior of real ants in the early 1990s [4]. In the AS, the experiences from previous attempts in solution searching guide artificial ants to construct feasible solutions and proceeds in a cooperative manner. Many researchers have proposed new approaches based on the preceding description of design principles which are known as Ant Colony Optimization (ACO), such as the ant colony system (ACS) [5] and the $MAX$-$MIN$ Ant System ($MMAS$) [6].

In general, every metaheuristic algorithm must address two major capabilities for a search space: exploration and exploitation [7-8]. Exploration is a process of discovering potential solutions by directing the search space to entirely new regions to search for better solutions. To retain visited promising solutions, exploitation is a process of utilizing such visited information in obtaining areas to determine which regions of the search space should be explored next. The exploitation capability often suffers from a loss of diversity in feasible solutions, thus increasing the risk of becoming trapped in local optima. The advantage of the exploration is that it has higher opportunity of hopping from one local optimum to another; concurrently, however, it greatly increases the risk of the metaheuristic algorithm being unable to converge. For example, ant colony optimization can use the construction information of past solutions to ensure the validity of the generated problem solutions. This also leads the exploitation capability to become higher with the accumulation of search experiences, increasing the risk of the algorithm falls into local optimum as the search time increases. We can know that the control mechanisms for balancing exploration and exploitation have gradually become an essential factor in improving the performance of metaheuristic algorithms [7-8].

Based on the design principle of balancing exploration and exploitation, we proposed $MMAS_{DRW}$-$SRP$ approach to improve the performance by taking proposed dynamic roulette wheel strategy.

The remainder of this paper is organized as follows. In the next section we describe the general formulation of the software release planning. In Section 3 we present the proposed $MMAS_{DRW}$-$SRP$ algorithm in detail. We also describe our experimental study and its results in Section 4. Finally, we conclude this paper in the last section.

2. Problem Formulation

The goal of release planning in incremental software development is to identify an optimal plan that maximizes the sum of all (weighted) priorities of all the different stakeholders. In this study, the symbol $F = \{f_1, f_2, \ldots, f_{|F|}\}$ refers to the set of features in a software project and $R = \{r_1, r_2, \ldots, r_{|R|}\}$ represents the set of releases that will be developed. The size of $F$ and $R$ are symbolized as $|F|$ and $|R|$, respectively.
It indicates that the feature $f_i$ has been assigned to release $r_j$ in the case of $x(i)=r_j$.

Let $P$ and $C$ be a set of the precedence and the coupling relationships between features, respectively. $(f_i, f_j) \in P$ indicates that a precedence relationship exists between $f_i$ and $f_j$, such that $f_j$ should not be delivered before $f_i$. Moreover, $(f_i, f_j) \in C$ indicates that a coupling relationship exists between $f_i$ and $f_j$, such that $f_i$ and $f_j$ should be delivered simultaneously. Precedence and coupling relationship constraints are given as follows:

$$x(i) \leq x(j), \forall (f_i, f_j) \in P ,$$

$$x(i) = x(j), \forall (f_i, f_j) \in C .$$

Let $T = \{t_1, t_2, \ldots, t_m\}$ be the set of $|T|$ resource types in a software project. When delivering the feature $f_i$, the symbol $n_{ij}$ indicates the consumption amount of resource type $t_k$.

In addition, the symbol $\text{cap}_{t_k}$ denotes the maximum available amount of the resource $t_k$ in the release $r_j$. Thus, each release planning solution $x$ must satisfy the following resource bounded constraint:

$$\sum_{i \in S_{t_k}} n_{ij} \leq \text{cap}_{t_k}, \forall t_k \in T \text{ and } \forall r_j \in R .$$

Let set $S = \{s_1, s_2, \ldots, s_N\}$ be the collection of stakeholders in the software project. Each stakeholder $s_p \in S$ is assigned a relative importance $\lambda_p \in \{1, 2, \ldots, 9\}$ by the project manager. The symbol $\text{value}_{ij} \in \{1, 2, \ldots, 9\}$ represents the perceived value of the feature $f_i$ for stakeholder $s_p$. The symbol $\text{urgency}_{ij} \in \{1, 2, \ldots, 9\}$ indicates the satisfaction with the situation that the feature $f_i$ is assigned to release $r_j$ for the stakeholder $s_p$. Ruhe et al. proposed the following objective function $F(x)$ for the feasible solution $x$:

$$F(x) = \sum_{i, j} \sum_{s \in S} \text{WAS}_{s} ,$$

where $\text{WAS}_{s}$ refers to the weighted average satisfaction (WAS) of each stakeholder priorities for all features $f_i$ when assigned to release $r_j$ as designated in the function:

$$\text{WAS}_{s} = \xi_s \times \left[ \sum_{i, s} \lambda_p \times \text{value}_{ij} \times \text{urgency}_{ij} \right] ,$$

where $\xi_s$ refers to the importance of the release $r_i$ in the software project development. Table 1 shows a software project example of software release planning. In this example, the maximum available amounts of the resource type $t_1$ in the release $r_i$ is $\text{cap}_{t_1}=1,300$. The $\text{urgency}_{ij}=(9, 0, 0)$ represents the degree of preference of assigning the feature $f_i$ in release 1, 2, and 3 is 9, 0, and 0 by the stakeholder $s_1$, respectively.

The dependency relationships between features in software projects can generally be characterized by a dependency graph. However, stakeholders usually only define direct relationships between features, and do not describe the derived relationships from current relation conditions. This increases the risk of the metaheuristic algorithms generating illegal solutions. To reduce the risk of generating illegal solutions, this study uses an activity graph $G=(V, E)$ to describe the direct and derived dependency relationships between features. In the activity graph, the set $V = \{a_1, a_2, \ldots, a_N\}$ consists of activities of software project and the set $E = \{e_{ij}\}$ consists of precedence relationships among activities. Each activity includes a single feature or several features with coupling relationships. The set $F^*$ refers to the collection of all features in activity $a_i$. The size of $V$ is symbolized as $|V|$, which represents the number of activities. A directed edge $e_{ij}$ in set $E$ indicates that a precedence relationship exists between activity $a_i$ and activity $a_j$. Figure 1 shows the activity graph of the problem instance described in Table 1.

![Fig.1. An example activity graph.](image)

### Table 1. Relevant information about the example project.

<table>
<thead>
<tr>
<th>$a_i$</th>
<th>$f_i$</th>
<th>Resource type $t_i$</th>
<th>Stakeholder $s_1$</th>
<th>Stakeholder $s_2$</th>
<th>Predecessor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>$f_1$</td>
<td>$t_{10}$ $t_{11}$ $t_{12}$</td>
<td>5,4,0 2, (0,3,6)</td>
<td>2, (0,3,6)</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$a_2$</td>
<td>$f_1$</td>
<td>$t_{10}$</td>
<td>5,4,0</td>
<td>$a_1$</td>
<td></td>
</tr>
<tr>
<td>$a_3$</td>
<td>$f_1$</td>
<td>$t_{10}$</td>
<td>5,4,0</td>
<td>$a_1$</td>
<td></td>
</tr>
<tr>
<td>$a_4$</td>
<td>$f_1$</td>
<td>$t_{10}$</td>
<td>5,4,0</td>
<td>$a_2$</td>
<td></td>
</tr>
<tr>
<td>$a_5$</td>
<td>$f_1$</td>
<td>$t_{10}$</td>
<td>5,4,0</td>
<td>$a_1$</td>
<td></td>
</tr>
<tr>
<td>$a_6$</td>
<td>$f_1$</td>
<td>$t_{10}$</td>
<td>5,4,0</td>
<td>$a_2$</td>
<td></td>
</tr>
<tr>
<td>$a_7$</td>
<td>$f_1$</td>
<td>$t_{10}$</td>
<td>5,4,0</td>
<td>$a_1$</td>
<td></td>
</tr>
<tr>
<td>$a_8$</td>
<td>$f_1$</td>
<td>$t_{10}$</td>
<td>5,4,0</td>
<td>$a_2$</td>
<td></td>
</tr>
<tr>
<td>$a_9$</td>
<td>$f_1$</td>
<td>$t_{10}$</td>
<td>5,4,0</td>
<td>$a_1$</td>
<td></td>
</tr>
<tr>
<td>$a_{10}$</td>
<td>$f_1$</td>
<td>$t_{10}$</td>
<td>5,4,0</td>
<td>$a_2$</td>
<td></td>
</tr>
</tbody>
</table>

3. The Proposed Algorithm

The MMAS\_DRW-SRP approach models the construction graph as a fully connected graph including $|V| \times |R|$ vertices to generate a feasible solution by artificial ant. Each vertex represents a pair between activities and releases, and any pair $(a_i, r_j)$ indicates assigning all features of activity $a_i$ to release $r_j$. A feasible solution can be formed as a tour with the length
The set refers to $\tau$, $\rho$, $\tau_w$, $W_{ij}$, $C$, and $F$ are a feature vector for each activity $a_i$, the resource bound constraints, and the quality of the feasible pair list $L$. The symbol $\rho$ indicates the probability for the candidate pair $(a_i, r_j)$ to be chosen, which can be defined as

$$P_y = \frac{W_{ij}}{\sum_{(a_i, r_j) \in C^*} W_{kl}}.$$  \hfill (10)

The selected target pair $(a_i, r_j)$ is then included in the feasible solution, and the target activity $a_i$ is removed from the sets $V_N$ and $V_R$. Repeat step 1 if $V_N$ is not empty; otherwise, proceed to the next step.

**Step 4.** Once an artificial ant completes the construction task of the feasible solution, the quality of the feasible pair list $L$ can be evaluated. The symbol $C_t$ refers to the evaluated value of the feasible pair list $L$, which can be defined as

$$C_t = \sum_{(a_i, r_j) \in L} W_{ij}.$$  \hfill (11)

After all artificial ants have completed the construction and evaluation of feasible solutions, the MMAS$_{DRW}$-SRP algorithm proceeds to the feedback phase. Otherwise, the artificial ants repeat the four steps in the construction phase.

### 3.3 Feedback phase of MMAS$_{DRW}$-SRP

In this phase, pheromone concentration is updated by

$$\tau_y = \begin{cases} (1-\rho)\tau_y + \rho\Delta\tau_y, & \text{if } (a_i, r_j) \in L_{GB} \\ (1-\rho)\tau_y, & \text{otherwise} \end{cases}$$  \hfill (12)

where $\rho$ is the evaporation rate, and $\Delta\tau_y$ is defined as

$$\Delta\tau_y = (|S|)^{-1} \times (|F|)^{-1} \times \log_{10} C_{\text{ref}}.$$  \hfill (13)

where $|S|$ and $|F|$ refer to the numbers of stakeholders and features in the software project and the symbol $C_{\text{ref}}$ refers to the quality of the global best solution $L_{GB}$ computed by (11). According to the limitations of pheromone concentration derived in the MMAS [6], the maximal and minimal amount of all possible pheromone trails are defined in the following:
4. Performance Study

The parameter values of these ACO-based approaches used for the test runs in all experiments [4, 5, 6, 10]. Due to the lack of a problem instance library of release planning in incremental software development, this study uses the problem instances based on the PSPLIB database of the multi-mode resource constrained project scheduling problem (MRCPSP). The benchmark sets for project instances of J10, J12, J14, J16, J18, and J20 include 536, 547, 551, 550, 552, and 554 problem instances, respectively. Each problem instance was carried out one solution-finding run. Each run was stopped after 10,000 feasible solutions had been evaluated.

A. The Derived Srp Problem Instance

For a project in MRCPSP, the set V consists of partially ordered activities and the set E is the set of precedence constraints among activities. Firstly, the content of feature set F, the precedence relationship set P, and the set of resource types T in SRP were set as the content of set V, the content of set E and the content of renewable resource types set R in MRCPSP, respectively. The coupling relationship set C was set as null in the derived problem instance. According to the setting situation of some parameters in literature [9], the content of set S, R and the values of the relative importance of stakeholders and releases were set as S=[s1, s2], R=[r1, r2, r3]={1, 2, 3}, λ1=6, λ2=4, ξ1=0.7, ξ2=0.3, ξ3=0.0, respectively. The calculation function of value_{pi}, n_{u} and Cap_{n} are defined as follow.

\[
value_{pi} = d_{ip}, \quad f_{i} \in F \text{ and } s_{p} \in S,
\]

\[
n_{u} = \sum_{m=1}^{M} \sum_{i} r_{int} \times (M_{i})^{-1}, \quad f_{i} \in F \text{ and } t \in T,
\]

\[
Cap_{n} = \left\{ \begin{array}{ll}
\sum_{i \in S} n_{i} \times (r_{i} + 1)^{-1}, & \text{if } 1 \leq r_{i} \leq 2 \text{ and } t = T; \\
0, & \text{if } r_{i} = 3.
\end{array} \right.
\]

For MRCPSP problem instance, the symbol d_{ip} indicates the duration of activity a_{i} in mode p, the symbol r_{int} indicates the consumption units of renewable resource t for activity a_{i} in mode m, and the symbol M_{i} indicates the number of execution mode for activity a_{i}. The urgency functions for stakeholder s_{1} and s_{2} are defined as follow.

\[
\text{urgency}_{s1} = \begin{cases} 
0, & \text{if } 1 \leq value_{pi} \leq 3; \\
(value_{pi}, 10 - value_{pi}), & \text{otherwise};
\end{cases}
\]

\[
\text{urgency}_{s2} = \begin{cases} 
0, & \text{if } 1 \leq w_{i} \leq 3; \\
(w_{i}, 10 - w_{i}), & \text{otherwise}.
\end{cases}
\]

where the definition of the symbol w_{i} is defined as follow

\[
w_{i} = (L_{i})^{2} \times |SUCC_{i}| \times (\max\{d_{i} \mid \forall a_{i} \in V\})^{-1} \times 10,
\]

where the symbol L_{i} indicates the maximal topology path length of activity a_{i} in MRCPSP problem instance, and the symbol |SUCC_{i}| represents the number of all immediate successors of activity a_{i} in MRCPSP problem instance. Obviously, an activity of MRCPSP problem instance with more successors and higher length of maximal topology path means that this activity has higher influence on software project, and thus for the corresponding feature in derived SRP problem instance was assigned in early release. The urgency function for stakeholder s_{2} defined in equation (20) is designed based on the characteristic of precedence constraints, and the urgency function for stakeholder s_{1} defined in equation (19) is designed based on the business value of features.

B. Performance Comparison

This subsection presents the experimental results obtained from the performance comparison of the proposed MMAS{\textsuperscript{DRW}}-SRP approach with other ACO-based approaches AS-SRP, AS\textsuperscript{Rank}-SRP, ACS-SRP, and MMAS-SRP. In this experiment, the percentage of optimal solution found was measured for each solution-finding run on the J10, J12, J14, J16, J18, and J20 benchmark test instances. Table 2 demonstrates the experimental results. The experimental results shown in Table 2 indicate that MMAS{\textsuperscript{DRW}}-SRP performs better than the other ACO-based approaches, especially when trying to solve the J20 benchmark. This is because the MMAS{\textsuperscript{DRW}}-SRP approach is capable of balancing the exploration and exploitation capabilities according to the search situation. For the traditional ant colony optimization technologies, the exploitation capability becomes higher with the accumulation of search experiences. Thus, the risk of the algorithm falling into local optimum increases over time.

<table>
<thead>
<tr>
<th></th>
<th>AS-SRP</th>
<th>AS\textsuperscript{Rank}-SRP</th>
<th>ACS-SRP</th>
<th>MMAS-SRP</th>
<th>This work</th>
</tr>
</thead>
<tbody>
<tr>
<td>J10</td>
<td>80.04%</td>
<td>84.33%</td>
<td>89.93%</td>
<td>95.52%</td>
<td>100.00%</td>
</tr>
<tr>
<td>J12</td>
<td>79.16%</td>
<td>84.28%</td>
<td>87.57%</td>
<td>95.25%</td>
<td>100.00%</td>
</tr>
<tr>
<td>J14</td>
<td>75.86%</td>
<td>79.49%</td>
<td>83.48%</td>
<td>94.56%</td>
<td>99.82%</td>
</tr>
<tr>
<td>J16</td>
<td>65.82%</td>
<td>68.36%</td>
<td>75.82%</td>
<td>86.73%</td>
<td>98.91%</td>
</tr>
<tr>
<td>J18</td>
<td>55.07%</td>
<td>63.95%</td>
<td>68.66%</td>
<td>84.24%</td>
<td>96.01%</td>
</tr>
<tr>
<td>J20</td>
<td>50.18%</td>
<td>59.57%</td>
<td>63.00%</td>
<td>80.14%</td>
<td>94.40%</td>
</tr>
</tbody>
</table>

To provide an intuitive explanation of what is happening during the search of the ACO-based approaches, this study analyzes the exploitation and exploration capabilities by tracking the evolution of the distance between ant paths along the search. The concept of distance is a way to calculate the difference between ant paths which was proposed in [11].
of pairs that appear in the feasible solution $L_1$, but not in the feasible solution $L_2$. A higher value of the average distance among all ants indicates that the algorithm has higher exploration capability. A smaller value of the average distance among all ants indicates that the exploitation capability of the algorithm has been promoted.

![Comparison of solution-finding behavior among ant-inspired search techniques.](image)

Figure 2 shows the value of the average distance among all ants in each iteration of five ACO-based algorithms for a problem instance with 20 activities. After iteration 10, the value of the average distance among all ants ranges between 0 and 2 for traditional ant colony optimization technologies. This indicates that the search is focusing more on exploitation after a short search of exploration. The traditional ant colony optimization technologies do not have enough exploration capability to help the algorithms jump out of the local optimal. For the MMAS$_{DRW}$-SRP approach, the value of the average distance among ants ranges between 1-3 and 6-8. This means that the curve of average distance for the MMAS$_{DRW}$-SRP approach exhibits extreme shakiness. When the average distance ranges between 1 and 3, the MMAS$_{DRW}$-SRP approach has more exploitation capability, and when the range of the average distance ranges between 6 and 8, the exploration capability of the MMAS$_{DRW}$-SRP approach has been improved. When the algorithm finds difficulties during the MMAS$_{DRW}$-SRP approach with more exploitation capability, the exploration capability has been improved to find the potential new solutions. When the algorithm discovers many local minima during the MMAS$_{DRW}$-SRP approach with more exploration capability, the MMAS$_{DRW}$-SRP approach facilitates more exploitation to improve current approximate optimal solutions based on past searching experiences. According to the experimental results in table 3, the automatic alternate shifting between exploitation and exploration is an essential feature of the MMAS$_{DRW}$-SRP approach for improving problem-solving performance.

5. Conclusion

Balancing between the exploration and exploitation capabilities is an essential task in improving the metaheuristic algorithm’s performance in recent years. The exploration capability of the traditional ant colony optimization technologies become weakness after accumulating great amount of search experiences. Therefore, we introduce a novel dynamic roulette strategy based on the MAX-MIN Ant System to balance exploration and exploitation capabilities to improve performance. The proposed MMAS$_{DRW}$-SRP algorithm dynamically decides the number of candidate pairs based on the searching situation to balance exploration and exploitation capabilities, and then improve problem-solving performance by reducing the risk of the algorithm falling into local optimal. The experimental results presented in this paper indicate that the proposed MMAS$_{DRW}$-SRP algorithm performs significantly better than other traditional ACO-based algorithms in terms of solution quality. It will be seen this that the dynamic roulette strategy can be able to improve the problem-solving performance.

Reference