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

Yu-Qing Huang[1], Chuan-Wen Chiang [2], Cheng-Hsu Huang[3]

[1]Department of Computer Science & Information Engineering, National Central University, Taoyuan County 32001, Taiwan
[2]Department of Computer and Communication Engineering, National Kaohsiung First University of Science and Technology, Nanzih, Kaohsiung 811, Taiwan, R.O.C.
[3]Department 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 *MM*AS$_{DRW}$-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 *MM*AS$_{DRW}$-SRP is demonstrated by comparing it against conventional ACO-based approaches. Experimental results indicate that the proposed *MM*AS$_{DRW}$-SRP algorithm performs significantly better than the competitive approaches.

Index Terms - software release planning (SRP), ant colony optimization (ACO), *MAX-MIN* Ant System (*MM*AS).

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 metahheuristic 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 (*MM*AS) [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 metachuristic algorithms [7-8]. Based on the design principle of balancing exploration and exploitation, we proposed *MM*AS$_{DRW}$-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 *MM*AS$_{DRW}$-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 $|R|$, representing the number of features and releases, 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:

$$x(i) \le x(j), \forall (f_i, f_j) \in P, \quad (1)$$

$$x(i) = x(j), \forall (f_i, f_j) \in C. \quad (2)$$

Let $T = \{t_1, t_2, \ldots, t_{|T|}\}$ be the set of $|T|$ resource types in a software project. When delivering the feature f_j, the symbol n_{jk} indicates the consumption amount of resource type t_k. In addition, the symbol cap_{ik} denotes the maximum available amounts of the resource t_k in the release r_i. Thus, each release planning solution x must satisfy the following resource bounded constraint:

$$\sum_{x(i)=r_j} n_{ik} \le cap_{jk}, \forall t_k \in T \text{ and } \forall r_j \in R. \quad (3)$$

Let set $S = \{s_1, s_2, \ldots, s_{|S|}\}$ 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 $value_{pi} \in \{1, 2, \ldots, 9\}$ represents the perceived value of the feature f_i for stakeholder s_p. The symbol $urgency_{pij} \in \{1, 2, \ldots, 9\}$ 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:

$$F(x) = \sum_{r_j \in R} \sum_{i:x(i)=r_j} WAS_{ij}, \quad (4)$$

where WAS_{ij} 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:

$$WAS_{ij} = \xi_j \times \left[\sum_{s_p \in S} \lambda_p \times value_{pi} \times urgency_{pij} \right], \quad (5)$$

where ξ_i 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_1 is cap_{11}=1,300. The $urgency_{19}$=(9, 0, 0) represents the degree of preference of assigning the feature f_9 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 $V = \{a_1, a_2, \ldots, a_{|V|}\}$ 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_i^+ 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.

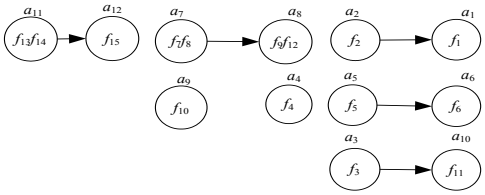

Fig.1. An example activity graph.

Table 1. Relevant information about the example project.

a_i	f_i	Resource type t_j				Stakeholder s_1		Stakeholder s_2		Predecessor
		t_1:n_{i1}	t_2:n_{i2}	t_3:n_{i3}	t_4:n_{i4}	$value_{1i}$	$urgency_{1i}$	$Value_{2i}$	$urgency_{2i}$	
a_1	f_1	150	120	20	1,000	6	(5,4,0)	2	(0,3,6)	a_2
a_2	f_2	75	10	8	200	7	(5,0,4)	5	(9,0,0)	-
a_3	f_3	400	100	20	200	9	(9,0,0)	3	(2,7,0)	-
a_4	f_4	450	100	40	0	5	(2,7,0)	7	(7,2,0)	-
a_5	f_5	100	500	40	0	3	(7,2,0)	2	(9,0,0)	-
a_6	f_6	200	400	25	25	9	(7,2,0)	3	(5,4,0)	a_5
a_7	f_7	50	250	20	500	5	(9,0,0)	3	(2,7,0)	-
	f_8	60	120	19	200	7	(8,1,0)	1	(0,0,9)	
a_8	f_9	280	150	40	1,500	6	(9,0,0)	5	(0,8,1)	a_7
	f_{12}	100	300	25	50	3	(9,0,0)	7	(0,6,3)	
a_9	f_{10}	200	300	40	500	2	(5,4,0)	1	(0,0,9)	-
a_{10}	f_{11}	250	375	50	150	1	(8,1,0)	5	(0,7,2)	a_3
a_{11}	f_{13}	100	250	20	50	7	(9,0,0)	9	(9,0,0)	-
	f_{14}	0	100	15	0	8	(9,0,0)	3	(6,3,0)	-
a_{12}	f_{15}	200	150	10	0	1	(0,0,9)	5	(3,6,0)	a_{11}
cap_{1j}		1,300	1,450	158	2,200					
cap_{2j}		1,046	1,300	65	1,750					

3. The Proposed Algorithm

The *MM*AS$_{DRW}$-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

$|V|$ in the construction graph. For instance, in the software project presented in Table 1, the pair list $L = <(a_1, 1), (a_2, 1), (a_3, 1), (a_4, 3), (a_5, 1), (a_6, 2), (a_7, 2), (a_8, 3), (a_9, 3), (a_{10}, 3), (a_{11}, 1), (a_{12}, 2)>$ is an example of feasible solutions, while the pair $(a_7, 2)$ indicates that the features f_7 and f_8 in the activity a_7 have been assigned to release 2. *MM*AS$_{DRW}$-SRP consists of initialization, construction, and feedback phases. The following subsections describe the operational processes of the *MM*AS$_{DRW}$-SRP approach.

A. Initialization phase of MMAS$_{DRW}$-SRP

The main task of this phase is to determine the initial values of the parameters employed in executing *MM*AS$_{DRW}$-SRP approach. For example, the initial value of the pheromone is set as 0.1 in this study.

B. Construction phase of MMAS$_{DRW}$-SRP

The construction phase constructs feasible solutions and then evaluates their quality. In the *MM*AS$_{DRW}$-SRP approach, each feasible solution is determined through four steps: (1) constructing the set of feasible pairs and calculating the weights of all feasible pairs; (2) constructing the set of candidate pairs; (3) selecting the target pair; and (4) evaluating the quality of the feasible solution. These four steps are detailed as follows.

Step 1. The set V_N consists of activities with an undetermined release. An activity that has not been assigned to any release is called a ready activity if it does not have any immediate predecessor activity or the releases of all its immediate predecessor activities have been assigned. According to the content of the feasible solution pair list L, the set of ready activities V_R is constructed based on the satisfactions of the precedence relationships. For each ready activity $a_i \in V_R$, according to the current content of the feasible pair list L, the set of legal release FR_i for the ready activity a_i has been constructed based on the satisfactions of the precedence relationships of activities and the resource bound constraints. After completing the construction of the legal release set for each ready activity, the set of feasible pair set F^* is constructed by the construction rule defined as

$$F^* = \{(a_i, r_j) \mid \forall a_i \in V_R \text{ and } \forall r_j \in FR_i\}. \quad (6)$$

For each pair $(a_i, r_j) \in F^*$, calculate the weight value w_{ij}, which indicates the suitability of ready activity a_i being assigned to release r_j. The weight value w_{ij} is given by

$$w_{ij} = [\tau_{ij}]^{\alpha} \times [\frac{WAS_{ij}^{+}}{\max\{WAS_{ij}^{+} \mid \forall r_j \in R\}}]^{\beta}, \quad (7)$$

where τ_{ij} represents the amount of pheromone trail for activity a_i to be assigned to release r_j, and two predefined parameters, α and β, are employed to adjust the relative influence of the pheromone trail and heuristic function. The two parameters are set as $\alpha = 1$ and $\beta = 2$. Moreover, WAS_{ij}^{+} refers to the weighted average satisfaction (WAS) when the activity a_i is assigned to the release r_j, which can be defined as

$$WAS_{ij}^{+} = \xi_j \times [\sum_{f_u \in F_i^*} \sum_{s_p \in S} \lambda_p \times value_{pu} \times urgency_{puij}]. \quad (8)$$

Step 2. The candidate pair set C^* consists of all the candidate pairs selected from the feasible pair F^* using random selection. This study proposes the dynamic strategy for constructing the candidate pair set C^*. The dynamic strategy is defined as

$$|C^*| = \min\{[\log_{|F|}(t_{notimprove} + 1)]^{-1} \times |F^*|, |F^*|\}, \quad (9)$$

where $t_{notimprove}$ refers to the number of unimproved iterations for the global best solution L_{GB}. After determining the number of candidate pairs, the candidate pair set is accomplished by selecting $|C^*|$ different pairs from feasible pair F^* through random selection.

Step 3. After constructing the candidate pair set, the artificial ant chooses the target pair (a_t, r_t) from the candidate pair set using roulette wheel strategy. The symbol p_{ij} indicates the probability for the candidate pair (a_i, r_j) to be chosen, which can be defined as

$$p_{ij} = \frac{w_{ij}}{\sum_{(a_k, r_l) \in C^*} w_{kl}}. \quad (10)$$

The selected target pair (a_t, r_t) is then included in the feasible solution L, and the target activity a_t 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_L refers to the evaluated value of the feasible pair list L, which can be defined as

$$C_L = \sum_{(a_i, r_j) \in L} WAS_{ij}^{+}. \quad (11)$$

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

3.3 Feedback phase of *MM*AS$_{DRW}$-SRP

In this phase, pheromone concentration is updated by

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

where ρ is the evaporation rate, and $\Delta\tau_{ij}$ is defined as

$$\Delta\tau_{ij} = (|S|)^{-1} \times (|F|)^{-1} \times \log_{|F|} C_{L_{GB}}, \quad (13)$$

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

$$\tau_{\max} = \frac{1}{\rho} \times (|S|)^{-1} \times (|F|)^{-1} \times \log_{|F|} C_{L_{GB}} \quad (14)$$

$$\tau_{\min} = \frac{\tau_{\max}}{2 \times |V|} \quad (15)$$

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 $S=\{s_1, s_2\}$, $R=\{r_1, r_2, r_3\}=\{1, 2, 3\}$, $\lambda_1=6$, $\lambda_2=4$, $\xi_1=0.7$, $\xi_2=0.3$, $\xi_3=0.0$, respectively. The calculation function of $value_{pi}$, n_{it} and Cap_{it} are defined as follow.

$$value_{pi} = d_{ip}, \quad f_i \in F \text{ and } s_p \in S, \quad (16)$$

$$n_{it} = \sum_{m=1}^{M_i} r_{imt} \times (M_i)^{-1}, \quad f_i \in F \text{ and } t \in T, \quad (17)$$

$$Cap_{it} = \begin{cases} \sum_{f_j \in F} n_{jt} \times (r_i + 1)^{-1}, & \text{if } 1 \le r_i \le 2 \text{ and } t \in T; \\ 0, & \text{if } r_i = 3. \end{cases} \quad (18)$$

For MRCPSP problem instance, the symbol d_{ip} indicates the duration of activity a_i in mode p, the symbol r_{imt} 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.

$$urgency_{1i} = \begin{cases} (0, value_{pi}, 10 - value_{pi}), & \text{if } 1 \le value_{pi} \le 3; \\ (value_{pi}, 10 - value_{pi}, 0), & \text{otherwise;} \end{cases} \quad (19)$$

$$urgency_{2i} = \begin{cases} (0, w_i, 10 - w_i), & \text{if } 1 \le w_i \le 3; \\ (w_i, 10 - w_i, 0), & \text{otherwise.} \end{cases} \quad (20)$$

where the definition of the symbol w_i is defined as follow

$$w_i = (L_i)^2 \times |SUCC_i| \times (\max\{d_i \mid \forall a_i \in V\})^{-1} \times 10, \quad (21)$$

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 successor 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 *MM*AS_{DRW}-SRP approach with other ACO-based approaches AS-SRP, AS_{Rank}-SRP, ACS-SRP, and *MM*AS-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 *MM*AS_{DRW}-SRP performs better than the other ACO-based approaches, especially when trying to solve the J20 benchmark. This is because the *MM*AS_{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 2. Percentage ratio of the number of optimal solutions found by *MM*AS_{DRW}-SRP and the competitive approaches.

	AS-SRP	**AS_{rank}-SRP**	**ACS-SRP**	***MM*AS-SRP**	**This work**
J10	80.04%	84.33%	89.93%	95.52%	100.00%
J12	79.16%	84.28%	87.57%	95.25%	100.00%
J14	75.86%	79.49%	83.48%	94.56%	99.82%
J16	65.82%	68.36%	75.82%	86.73%	98.91%
J18	55.07%	63.95%	68.66%	84.24%	96.01%
J20	50.18%	59.57%	63.00%	80.14%	94.40%

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]. For release planning, one method of measuring the distance *dist* (L_1, L_2) between two solutions L_1 and L_2 is to count the number

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 algorithm has been promoted.

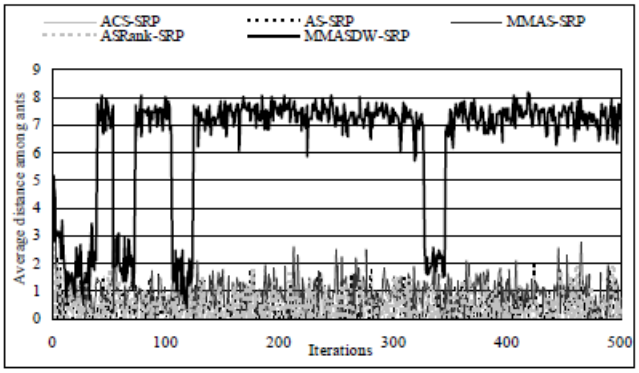

Fig. 2. Comparison of solution-finding behavior among ant-inspired search techniques.

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 the local optimal. For *MM*AS$_{DRW}$-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 *MM*AS$_{DRW}$-SRP approach exhibits extreme shakiness. When the average distance ranges between 1 and 3, the *MM*AS$_{DRW}$-SRP approach has more exploitation capability, and when the range of the average distance ranges between 6 and 8, the exploration capability of the *MM*AS$_{DRW}$-SRP approach has been improved. When the algorithm finds difficulties during the *MM*AS$_{DRW}$-SRP approach with more exploitation capability, the exploration capability has been improved to find the potential new solutions. When the algorithm discover many local minima during *MM*AS$_{DRW}$-SRP approach with more exploration capability, the *MM*AS$_{DRW}$-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 *MM*AS$_{DRW}$-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 *MM*AS$_{DRW}$-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 *MM*AS$_{DRW}$-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

[1] M. Ramzan, M. A. Iqbal, M. A. Jaffar, A. Rauf, S. Anwar and A. A. Shahid, "Project Scheduling Conflict Identification and Resolution using Genetic Algorithms," Proceeding of International Conference on Information Science and Applications (ICISA 2010), IEEE Press, 21-23 April 2010, pp. 1-6, doi: 10.1109/ICISA.2010.5480400.

[2] D. Greer and G. Ruhe, "Software release planning: an evolutionary and iterative approach," Journal of Information and Software Technology, Vol. 46, pp. 243-253 (2004)

[3] G. Ruhe and A. N.o The, "Hybrid Intelligence in Software Release Planning," International Journal of Hybrid Intelligent Systems, Vol. 1, issue 1-2, pp. 99-110 (2004)

[4] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a Colony of Cooperating Agents", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 26, no. 1, pp. 29-41, 1996.

[5] M. Dorigo and L. M. Gambardella "Ant Colony System: A cooperative learning approach to the traveling salesman problem," IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1, pp. 53-66 (1997)

[6] T. Stűtzle and H. H. Hoos, "*MAX-MIN* Ant System", Journal of Future Generation Computer Systems, Vol. 16, Issue 8, pp. 889-914 (2000)

[7] S.-H. Liu, M. Mernik and B. R. Bryant, "Entropy-Driven Parameter Control for Evolutionary Algorithms," Journal of Informatica, Vol. 31, No. 1, pp. 41-50 (2007)

[8] J. Zhao and H. Sun, "A Two Sub-swarm Exchange Particle Swarm Optimization Considering Exploration and Exploitation," Proceeding of International Conference on Industrial Mechatronics and Automation (ICIMA 2010), IEEE Press, 30-31 May 2010, pp. 530-533, doi: 10.1109/ICINDMA.2010.5538254.

[9] G. Ruhe and M. O. Saliu, "The Art and Science of Software Release Planning," Journal of IEEE Software, Vol. 22, Issue 6, pp. 47–53 (2005)

[10] B. Bullnheimer, R. F. Hartl and C. Strauss, "A New Rank Based Version of the Ant System: A Computational Study," Central European Journal for Operations Research and Economics, Vol. 7, No. 1, pp. 25-38 (1999)

[11] M. Dorigo and T. Stützle, Ant Colony Optimization, The MIT Press, Cambridge (2004)