International Journal of Computational Intelligence Systems, Vol. 6, No. 3 (May, 2013), 530-547

Resource-constraint Multi-project Scheduling with Priorities and Uncertain Activity Durations

Zheng Zheng[1,2+], Lin Shumin[2], Guo Ze[2], Zhu Yueni[2]

[1]*Science and Technology on Aircraft Control Laboratory, Beijing 100191, China*

[2]*School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China*

E-mail: zhengz@buaa.edu.cn

Received 28 May 2012

Accepted 21 November 2012

Abstract

Resource-constraint multi-project scheduling is one of the most important topic in the field of project management. Most current works solve this problem based on an idea that multiple projects can be simply emerged into a super-project in a deterministic environment, regardless of the project priority and robustness of schedules. This paper discusses the RCMPSP with priority and formulates a discrete bi-objective decision model. A modified NSGA-II based algorithm is presented to solve the model. Furthermore, we design systematic experiments to investigate the interrelationship between robustness and its related project parameters, including order strength, resource constrainedness and uncertainty level. The results demonstrate the effectiveness of the solution algorithm and show that the three parameters indeed have evident impacts on the robustness and makespan of projects.

Key Words: Multi-Project Scheduling, Priorities, Robustness, Uncertain Activity Durations, Uncertainty

1. Introduction

Resource-constraint multi-project scheduling problem (RCMPSP) comes from practical multi-project environments in which a number of projects concurrently share limited resources in precedence or other constraints. Contemporary organizations or enterprises organize work pervasively in multi-project environments as Payne [1] said, "Up to 90%, by value, of all projects are carried out in the multi-project context, and thus the impact of even a small improvement in their management on the project management field could be enormous". Although RCMPSP plays a vital role in project management, there are not much fruits on the topic as those on single project scheduling (i.e. resource-constraint project scheduling problem, RCPSP). The main reason comes from its high complexity, which is affected by many factors, such as the huge solution space, the intensely contending for resources, various and conflicting objectives, the inter-project dependence and priority, the high level of uncertainty and so on. Some of them are difficult to be handled or considered adequately in the characterization or the solution of the problem.

Current studies associated with RCMPSP mostly concentrate on its solution algorithms. A RCMPSP belongs to NP-hard problems, thus exact methods could hardly deal with it. As a result, researchers proposed different meta-heuristic to solve RCMPSP. Genetic algorithms (GA) are typically used methods. Kim [2] proposes a hybrid genetic algorithm with fuzzy logic controller to minimize total project time and to minimize total tardiness penalty. Yassine [3] utilizes an algorithm, called competent GA (CGA), which enhanced traditional GA by adding a local strategy to solve RCMPSP. Experiments show that CGA outperforms many other priority-rule-based heuristics. Goncalves [4] presents a genetic algorithm based on random key representation, and a schedule generation creating parameterized active schedules. Other kinds of algorithms for solving RCMPSP include priority rules based methods [5-8] and hybrid heuristic approaches [9-11].

Although successful attempts have been made for the solution of RCMPSP, there still remain many issues to be investigated. First, current approaches for general RCMPSP seem not suitable for certain practical use, since in real-world project management, there exist

typical scenarios in which each project has its own priority. For example, in the task scheduling problem for satellite testing, different tasks usually have different priorities according to the importance degrees of test targets (For example, satellites or systems). To model the problems more accurately, task schedulers should adopt priority strategies to arrange the order of executing projects. To be specifically, projects with high priority should get resources prior to those with low priority. We called this problem as resource-constraint multi-project scheduling problem with priorities (RCMPSPP). Most existing works solve RCMPSP by emerging all projects into a virtual big project, that is, the single project scheduling approach. However, this approach loses its effectiveness in RCMPSPP. If the priorities are taken into consideration, projects cannot be emerged into a big project as before since they are constrained by certain orders. Second, uncertainty has not been considered sufficiently when planning projects. Current works usually solve the problem with the only objective of minimizing the total project makespan, ignoring the uncertainty of projects. In realistic dynamic environments where the conditions often change unexpectedly with the lapse of time, the uncertainty is the main factor that leads to the interruption of schedule execution frequently. According to Fox and Ringer's survey [15], only less than 5% of the time spent in practice on scheduling is for developing new schedules, while 95% of the time is spent revising and maintaining schedules based on daily progress and changing assumptions. The uncertainties may stem from a number of possible sources. For example, activities may take more or less time than originally estimated, resources may become unavailable, material may arrive behind schedule, finish times and due dates may have to be changed, and etc [16]. A slight initial delay may ripple spread across all the projects, and thus results in prodigious delay because of the large network, inter-project dependence and contending for resources, etc.

This research intends to study the baseline schedule of RCMPSPP with uncertain activity durations. To precisely describe the uncertainty, we utilize the concept of robustness, which is used to measure the ability of a schedule to absorb uncertainty or to suppress the propagation of disruptions to keep it stable. A baseline schedule is utilized to serve as an estimation of the overall project scope and a yardstick of resource allocation, external planning, adjusting or rescheduling afterward. It is usually established by assuming deterministic information on resource usage and activity durations in the light of expectations or experiences. The more robust the baseline schedule is, the more precise the estimation is, and the less cost of delay and adjustment of the schedule exist in real execution. To build a robust baseline schedule, Van De Vonder et al. [17, 18] and Lambrechts et al. [19, 20] have developed several proactive heuristic or meta-heuristic procedures to protect the schedule from future interruption with a trade-off between robustness and makespan. However, the procedures cannot be applied to RCMPSPP directly because the impacts of specific characteristics such as project structures and uncertainty levels are not considered adequately. Therefore, new procedures need to be proposed, in which the systematical analysis of the impacts is necessary. This is a research topic of this paper.

According to the above analysis, the main work of this paper is threefold. Firstly, a conceptual model for RCMPSPP is presented, which includes two objectives to optimize the makespan and the robustness measure simultaneously. Secondly, a solution algorithm for RCMPSPP is proposed, which is a bi-objective genetic algorithm based on the NSGA-II approach. Thirdly, we further analyze the robustness of solutions obtained from the proposed algorithm and try to answer the following questions: (1) What are the effects of three related project parameters on robustness and makespan? (2) What are the relationship between the two objectives, i.e. robustness and makespan? These questions are important both in practice and in theory. It can help the decision makers to determine the right schedules when they confront the dilemma of comprising between robustness and makespan. What's more, the answers can give us a view of how parameters affect the solutions, and help researchers adjust them more properly so as to improve the performance of solutions.

The remainder of the paper is organized as follows. Section 2 describes the resource-constraint multi-project scheduling problem with priorities and its conceptual model. Section 3 presents in detail an algorithm to solve the RCMPSPP based on the NSGA-II approach. Section 4 shows the experiments to justify the effectiveness of the algorithm and the impacts of parameters on its robustness and makespan. In Section 5, results and discussion are presented. Finally, the conclusion is given in Section 6.

2. RCMPSPP model and its analysis

2.1. *Problem description and conceptual model*

Resource-constraint multi-project scheduling problem with priorities (RCMPSPP) aims at finding a schedule that fixes start times and end times of activities, while

minimizing or maximizing one or more performance measures. Before the introduction of the conceptual model, parameters and constraints are presented in the following.

- $\mathcal{L} = \{1, \cdots, L\}$ denotes the set of L single projects with different priorities.
- Project $i_1 \in \mathcal{L}$ has a higher priority ω_1 than project $i_2 \in \mathcal{L}$. If $\omega_1 > \omega_2$, project i_1 has precedence over project i_2 to get resources. Nevertheless, preemption is not allowed.
- Each project $i \in \mathcal{L}$ consists of N_i interrelated activities in set $\mathcal{N}_i = \{1, \cdots, N_i\}$, where activities 1 and N_i are dummy activities, representing the start and the end of project i respectively.
- All the projects share K renewable resources in set $\mathcal{K} = \{1, \cdots, K\}$, and every resource $k \in \mathcal{K}$ has a constant amount of R_k units available at any time.
- The activities are subject to two kinds of constraints:
  (1) Precedence constraints. Each activity $j \in \mathcal{N}_i$ in project $i \in \mathcal{L}$ cannot be scheduled until all of its predecessor activities in set $\mathcal{P}_{ij} \subset \mathcal{N}_i$ are completed;
  (2) Resource constraints. Each activity $j \in \mathcal{N}_i$ requires r_{ijk} units of resource $k \in \mathcal{K}$ during its duration d_{ij}. The dummy activities require no resources and their durations are zero, and other activities durations are uncertain. All parameters are non-negative.
- The start times and end times of activities are fixed, which are $\boldsymbol{S} = (s_{11}, \cdots, s_{1N_1}, \cdots, s_{L1}, \cdots, s_{LN_L})$ and $\boldsymbol{F} = (f_{11}, \cdots, f_{1N_1}, \cdots, f_{L1}, \cdots, f_{LN_L})$ respectively.
- $I_t = \{(i,j) | s_{ij} \le t, f_{ij} > t\}$ represents the set of activities being processed at time instant t.

Thus, the conceptual model of RCMPSPP can be formalized as follows.

$$\min \quad \Phi(F) \tag{1}$$

$$\text{s.t.} \quad s_{ij} \ge f_{ij'}, i = 1,...,L; j = 1,...,N_i; j' \in P_{ij} \tag{2}$$

$$\sum_{(i,j)\in I_t} r_{ijk} \le R_k, i = 1,...L; j = 1,...,N_i; k = 1,...,K \tag{3}$$

$$f_{ij} = s_{ij} + d_{ij}, i = 1,...,L; j = 1,...,N_i \tag{4}$$

$$s_{ij} \ge 0, i = 1,...,L; j = 1,...,N_i \tag{5}$$

$$\omega_i > \omega_{i+1}, i = 1,...,L \tag{6}$$

Function vector (1) contains the objectives to optimize the performance measure. In this paper, $\boldsymbol{\Phi}(\boldsymbol{F})$ consists of two objectives, the minimal robustness measure and the shortest makespan. We will have an in-depth discussion about them in the Section 2.2 and 2.3. Formula (2) represents the precedence relationship between activities. Formula (3) makes the schedule satisfy the resource constraints at any time. Formula (4) shows the relation between start time and end time of an activity. Formula (5) forces the start time and end time to be non-negative. Formula (6) imposes priority relations between projects.

2.2 *Objectives for the model*

In project management, managers usually seek the schedule that can be implemented as what is planned. In other words, schedules are expected to be as stable as they can, especially for short ones, because a short schedule may cause a great adjustment cost. Hence, a robust baseline schedule is of great importance. Many delays or failures of project execution are attributed to the unexpected increase of activity durations in practice. Al-Fawzan and Haourai [21] view the deviation of activity duration as an important factor affecting a schedule, and develop the concept of robustness. Robustness of a schedule means the ability to cope with small increases in some activity durations caused by uncontrollable factors. Van de Vonder et al. [17] further distinguish the difference between quality robustness and solution robustness. Quality robustness is measured in terms of project duration, generally defined as the difference between the planed and the realized makespan. Solution robustness is defined as the function of the deviations between the planned and realized start times of activities.

In multi-project scheduling, because of the large number of activities and the propagation of delays, an increase in duration of one single activity can lead to delays of many activities. In this case, solution robustness is too rigorous for measuring the stability of start times of activities, while quality robustness is more appropriate for multi-project scheduling. Moreover, managers usually consider that the cost of delaying the completion times of projects preponderates over that of deviating from start times of the planned activities. Thus, in this paper we mainly concentrate on the quality robustness of RCMPSPP. Unfortunately, similar with the solution robustness, the quality robustness is also not easy to calculate. Theoretically, in order to calculate the robustness, it is necessary to know a priori disruption scenarios with their probability information and the parameters for all possible cases. However, it is unrealistic for the following reasons. First, these scenarios are usually not easy to know or describe beforehand. Second, the probability information is also not easy to obtain. Third, even if we can obtain the scenarios, the number is usually very huge in real-life project management, especially in multi-project environments. In this case, the burden of computing the robustness measure can be very heavy due to the combinatorial nature of project scheduling. Therefore, to measure the quality

robustness, a reasonable strategy is to limit the number of disruptions in the real project execution.

According to the above discussion, an assumption is made in this paper for the robustness measure: in every project execution, there is only one disruption resulting from the increase of duration of a single activity. Herroelen et al. [22] told that this assumption actually does not preclude more or less disturbances taking place during project execution, while the underlying idea is that disturbances are sufficiently sparse and spread over time and throughout the project network. Therefore, we can assume that the effect of one disturbance will not interact with the effects of another. The single disruption strategy can serve as a basis of analyzing the robustness for general cases.

During the execution of a project, once a disruption occurs, it is likely to cause frequently reallocation of resources and delays of projects. Therefore, we can use delays of projects to measure the robustness. To describe the robustness measure more clearly, some notations are presented first in the following.

- Δd_{ij}: the duration increment of activity j in project i, and $\Delta d_{ij} > 0$;
- ΔD_{lij}: the delay of project l caused by Δd_{ij} when schedule S is rescheduled. Note, if project l starts earlier than project i, $\Delta D_{lij} = 0$;
- $s_{lmax} = \max_{j=2,\cdots,N_l-1}\{s_{lj}\}$: the latest start time of activities of the project l, except for dummy activities;
- A_l: the set of activities which may affect the finish time of project l in schedule S, that is

$$A_l = \{(i,j)|i = l; j = 2,\cdots,N_l - 1\} \cup \{(i,j)|f_{ij} \leq s_{lmax}; i \neq l; j = 2,\cdots,N_l - 1\}.$$

Based on these, the robustness measure RM can be defined as

$$\text{RM} = \sum_{l=1}^{L} \frac{1}{|A_l|} \sum_{(i,j)\in A_l} \Delta D_{lij} \qquad (7)$$

under the condition that there is only one activity whose duration increases unexpectedly in a project execution. RM is equal to the sum of average delays of all projects. Obviously, the smaller the value of RM is, the better the robustness is. Note that RM depends on the rescheduling strategy, because different rescheduling strategies can generate different values of ΔD_{lij}.

On the other hand, a makespan is commonly viewed as the most important performance measure. It can be defined as

$$\text{makespan} = \max_{i=1,\cdots,L}\{f_{iN_i}\} \qquad (8)$$

where f_{iN_i} denotes the finish time of the last activity N in project i.

Now we establish two performance measures: robustness measure RM (Refer to Eq.7) and makespan (Refer to Eq.8). This forms a bi-objective conceptual model of RCMPSPP with objectives of minimizing the makespan measure and the robustness measure simultaneously. Due to its NP-hard property, we use multi-objective genetic algorithm to solve this problem in Section 3.

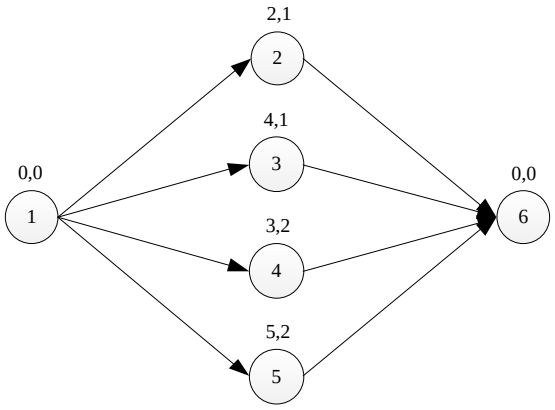


Fig.1. Sample project network

Fig.1 shows the network graph of a sample project. Each node represents an activity. Arrows in the figure represent the precedence relationship between activities. In the example, there is only one type of resource whose available capacities are 4 units. The numbers above a node are its duration and resource requirement. Fig.2 and Fig.3 are two different schedules to a multi-project network, composed of two projects with the same network as Fig.1. The priority of project 1 is higher than project 2. The two schedules have the same makespan of 12 units. However, they may have different RM values. If we give an increment of 1 unit to the duration of every activity in turn and reschedule the rest of posterior activities according to their start times in ascending order, RM of schedule 1 is $2/4 + 8/8 = 1.5$, and $2/4 + 5/8 = 1.125$ for schedule 2. This result illustrates that schedule 2 is more robust than schedule 1, though they have the same makespan.

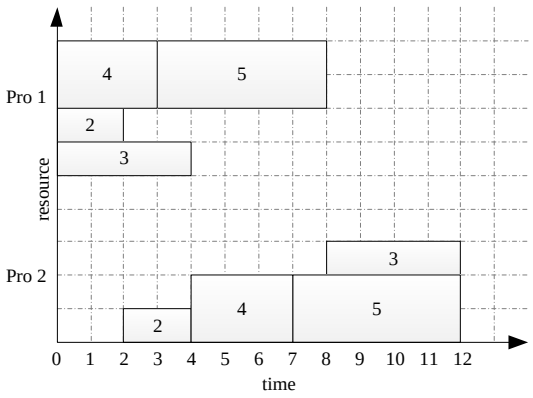


Fig.2. Schedule 1 for a two-project network

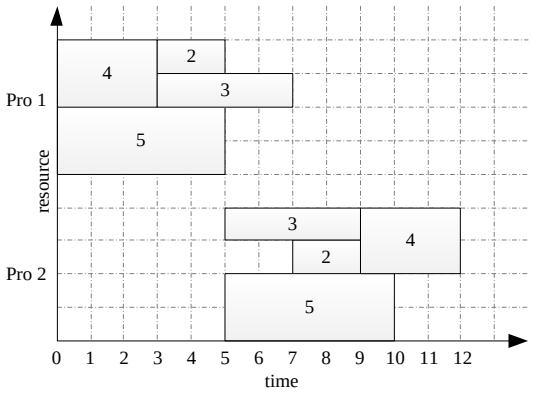


Fig.3. Schedule 2 for a two-project network

2.3. *Project parameters affecting robustness*

The schedule results are constrained by uncertainty, precedence constraints and resource constraints. Meanwhile, for the same group of projects, different schedules may result in different robustness. Thus, to express the strength of the three constraints that can influence robustness indirectly, we introduce corresponding typical parameters in the following.

(i) Uncertainty Level

The definition of RM above implies that the robustness relates to Δd_{ij}. In some sense, Δd_{ij} measures the uncertainty existing in durations of activities. To the uniform description, we define the uncertainty level as follows.

$$\mathrm{UL}_{ij} = \frac{\Delta d_{ij}}{d_{ij}} \tag{9}$$

Although Δd_{ij} for activities may be different, the uncertainty level of activities can be the same.

The uncertainty of project may spread across all the projects. It is caused by the constraints between activities that determine the resistance of project to the uncertainty (i.e. robustness). In a RCMPSPP, activities face two kinds of constraints: precedence constraints and resource constraints, which depend on activity characteristics and resource characteristics, respectively. Concerning the constraints, it is improper to ignore order strength and resource constrainedness, since they are essential factors that influence the constraints most.

(ii) Order strength

Order strength, denoted as OS, measures the complexity of the project network topology[23]. It is formulated as

$$OS = \frac{n_{pr}}{n(n-1)/2} \tag{10}$$

where n_{pr} is the number of precedence relations, including the transitive ones, and $n(n-1)/2$ denotes the theoretical maximum number of precedence relations.

(iii) Resource constrainedness

Resource constrainedness, denoted as RC, measures the levels of resource availability [23]. It is defined as

$$\mathrm{RC}_k = \frac{\bar{r}_k}{a_k} \tag{11}$$

where a_k denotes the total availability of renewable resource type k, $\bar{r}_k$ denotes the average quantity of resource type k required, formulated as

$$\bar{r}_k = \sum_{i=1}^{n} r_{ik} \Big/ \sum_{i=1}^{n} m, m = \begin{cases} 1, & r_{ik} > 0 \\ 0, & r_{ik} = 0 \end{cases}$$

In order to show the impact of these two parameters, a toy case with three projects is given here with its basic information shown in Table 1.

Table 1. Basic information of projects

Basic Information	Project 1	Project 2	Project 3
No. of activities	10	10	10
No. of resource type	4	4	4
Network structure	Network 1	Network 1	Network 2
Resource requirement	Requirement 1	Requirement 2	Requirement 1

In Table 1, the activity number and resource type of each project is fixed. Project 1 and Project 2 have the same

network structure but different requirement. Project 1 and Project 3 with the same requirement have different network structures. The scheduling results are shown in Fig.4 to Fig.6. It is evident that different network structures and resource requirements lead to completely different schedules (the makespans of each projects are 27, 56 and 36). Besides, the robustness of projects also varies from each other. Therefore, the makespan and robustness are quite dependent on network structures and resource requirements.

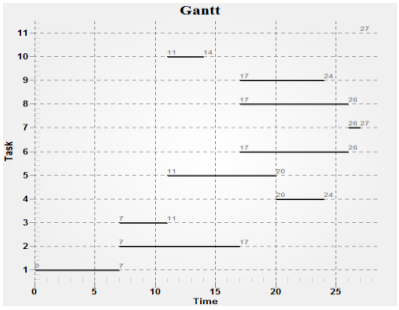


Fig.4. Schedule of Project 1(OS=0.3 RC=0.3)

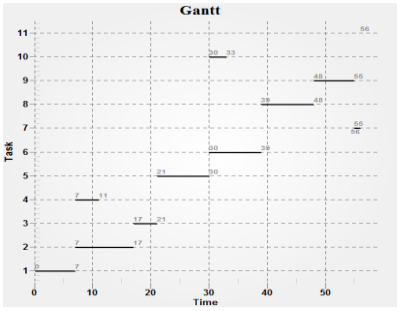


Fig.5 Schedule of Project 2(OS=0.3 RC=0.7)

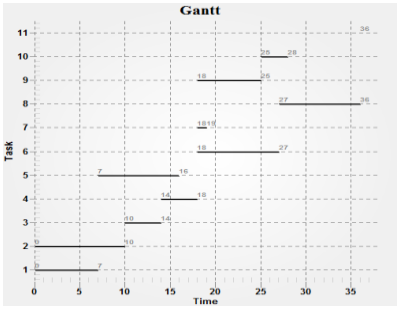


Fig.6 Schedule of Project 3(OS=0.7 RC=0.3)

3. A NSGA-II based solution algorithm for RCMPSPP

For multi-objective optimal problems, evolutionary algorithm (EA) is regarded as one of the most effective solution algorithms. Many EAs are proposed and studied in the last decades. Among them, NSGA-II [24] is one of the most well-known and efficient approaches. In this research, a NSGA-II based algorithm is proposed to solve the bi-objective model of RCMPSPP, achieving the proximity of Pareto optimal solutions. Fig. 7 shows the basic structure of NSGA-II based approach. Both the size of initial population P_0 and its offspring population Q_0 are N. The combined population R_t of size $2N$ is sorted out into $F = (F_1, F_2, \cdots)$ according to different non-domination ranks of solutions in R_t. The crowding-distance is used here to get an estimation of the density of solutions with the same non-domination rank. Define the crowded-comparison-operator $\prec_n$ as

$$i \prec_n j \text{ if and only if } (i_{rank} < j_{rank}) \text{ or } \big((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance})\big).$$

The next generation P_{t+1} is derived from the best N solutions in F. Q_{t+1} is generated from P_{t+1} by binary tournament selection, crossover and mutation. In tournament selection, $\prec_n$ is used to compare solutions. Obviously, NSGA-II holds the elite strategy through the combination of two populations to increase its performance. In the following, we will introduce the approach in detail by three parts.

NSGA-II based solution algorithm

Create initial population P_0, and its offspring population Q_0 by genetic operators;
$t = 0$, $R_0 = \emptyset$;
While not satisfy the stop criteria
  $R_t = P_t \cup Q_t$;
  $F =$ Fast-non-dominated-sort(R_t) ;
  $P_{t+1} = \emptyset$ and $i = 1$;
  While $|P_{t+1}| + |F_i| \leq N$
    Crowding-distance-assignment(F_i) ;
    $P_{t+1} = P_{t+1} \cup F_i$;
    $i = i + 1$;
  End While
  Sort F_i in descending order using $\prec_n$ operator ;
  $P_{t+1} = P_{t+1} \cup F_i[1:(1 - |P_{t+1}|)]$;
  Generate Q_{t+1} from P_{t+1} by genetic operators ;
  $t = t + 1$;
End While

Fig.7 The basic structure of NSGA-II based solution algorithm

3.1. *Chromosome representation*

Each chromosome is encoded as an activity sequence and divided into several segments. Every segment represents

a set of activities belonging to the same project, and there is no overlap among them. The positions of segments in the chromosome stand for their priorities. A chromosome of L projects with priorities in descending order can be encoded as follows:

$$\pi = (\underbrace{A_{11}, \cdots, A_{1N_1}}_{Project\ 1}, \cdots, \underbrace{A_{L1}, \cdots, A_{LN_L}}_{project\ L})$$

Here, we assume that the activity sequence is a precedence feasible permutation, and $\{A_{i1}, \cdots, A_{iN_i}\} = \{1, \cdots, N_i\}$ for $i = 1, \cdots, L$.

According to this representation, if we decode the chromosome from left to right, it can just satisfy the precedence of activities and the priority relationship of projects. Moreover, this form of encoding can remarkably reduce the searching space compared with merging all projects into a super-project, because it limits the range of permutation of activities in their own segments.

The initial population should be diverse enough to reduce the probability of falling into local optimum. It could be generated at random, according to priority rules, or by combining both of them. However, the priority relationship of projects and the precedence constraints of activities in the same project should be maintained. The initial population is generated according to the following criteria:

(i) Fix the dummy start activity of project 1 in the first position;
(ii) Activities are selected from the feasible set of activities whose predecessors have been fixed in the chromosome. The selection can be at random or through priority rules;
(iii) When a dummy end activity is fixed, the dummy start activity of the next project is the next activity.

3.2. *Schedule generation procedure*

A schedule generation procedure is to construct schedules from chromosomes. In this paper, we propose a procedure fit for the encoding in section 3.1 to generate active schedules. An active schedule is a feasible one in which an activity cannot start earlier if others do not delay. The basic idea of this procedure is that the activities represented by a chromosome are scheduled at their earliest times (ET) in turn from left to right under resource constraints, and once a dummy start activity is encountered, its start time is fixed to zero. This means that every project starts at zero. This strategy is called as ET strategy in the following. Clearly, the schedule generation procedure can satisfy the priority relationship among projects. Before the description of the procedure, we propose a property of ET strategy at first.

Property 1: *According to ET strategy, the activities except dummy start activities can only start at the finish times of other activities.*

Proof. We can prove the property with apagoge. Assume that an activity A of a feasible schedule starts at time t, but other activities don't finish at t. Then we can derive that none activities would start or finish during the interval(t', t), where

$$t' = max\{f_{ij} | f_{ij} < t, i = 1, \cdots, L, j = 1, \cdots, N_i\}$$

In other words, the remaining amount of each type of resources does not change during this time interval until activity A starts. This reveals that the amounts of remaining resources during $[t', t)$ is no less than those at time instant t when the amounts are enough for activity A. So activity A can start earlier at time t'. This contradicts ET strategy. □

Fig.8 illustrates the pseudo-code of the schedule generation procedure. Let T be the time sequence in ascending order, and T_g the g^{th} element. T is composed of finish times of all activities scheduled at present. Therefore, the start time of an activity can be selected from T. An activity A can start at t if and only if it satisfies resource and precedence constraints during $[t, t + d]$.

Schedule generation procedure

$T = (0)$
While $i \leq L$
  $s_{i1} = 0, f_{i1} = 0, j = 2$;
  While $j \leq N_i$
    Compute the latest finish time t' of predecessors of A_{ij}, i.e. $t' = \max\{f_{ij'} | i = L, j' \in \mathcal{P}_{ij}\}$;
    Locate the position g of t' in T;
    While not satisfy resource requirement of A_{ij} in $[T_g, T_g + d_{ij}]$
      $g = g + 1$;
    End While
    $s_{ij} = T_g, f_{ij} = T_g + d_{ij}$;
    If $f_{ij} \notin T$
      $T = (T, f_{ij})$;
      Sort T in ascending order;
    EndIf
    $j = j + 1$;
  End While
  $i = i + 1$;
End While

Fig.8 Schedule generation procedure

The proposed schedule generation procedure is an injection from chromosomes to schedules, that is, each chromosome corresponds to a unique schedule. Nevertheless, the encoding forms a multivalued mapping from schedules to chromosomes. A schedule may be encoded by more than one chromosome. The multivalued mapping actually relates to resource levels and complexity of projects network in project scheduling. Generally, if resources are scarce and the complexity of multi-project network is high, the multivalued mapping approximates to injection. Besides, multivalued phenomenon is more conspicuous with the increasing of resources and decreasing of network complexity. This phenomenon probably can have impacts on the efficiency of genetic operators, especially mutation, because it may lead genetic operations to take the same results.

3.3. *Genetic operators*

Genetic operators are applied in EAs to produce new populations and improve the quality of solutions. To survive the fittest, a selection operator is designed to determine which individuals can reproduce offspring, and which should die out. A crossover operator is the main way of creating new individuals from those selected parents individuals. In addition, a mutation operator is used to introduce randomness into evolution of EAs to increase searching areas, avoiding to fall into local optimum. In the following, the operators used in NSGA-II based algorithm will be introduced in detail.

3.3.1 *Selection* and *Mutation*

In NSGA-II, there are two selection operations in every single run. The first is to select P_{t+1} from combined population R_t. The best N individuals will be directly copied to P_{t+1}. It is called as elite strategy. This strategy can improve the quality of solutions from generation to generation. The second is to select Q_{t+1} from P_{t+1}. The 2-tournament selection mechanism is adopted this time. Two individuals are selected randomly from the population P_{t+1}, and the better one is preserved by comparing them using crowded-comparison-operator $\prec_n$. The selection pressure of the 2-tournament mechanism is relatively light, and this is helpful to increase the diversity of a population.

The mutation is used to avoid premature convergence of the population usually with very small probability at each generation. The mutation of an activity selected is performed as follows. At first, determine the position r_1 of its nearest predecessor and the position r_2 of its nearest successor. Then a random integer $r(r_1 < r \leq r_2)$ is generated as the position the activity is inserted in. The activity mutated cannot be dummy activity. Obviously, the mutation does not break the precedence and priority relationship as well.

3.3.2 *Crossover*

The crossover operator is similar to the traditional one-point crossover with the difference that a virtual precedence relationship is set between the dummy end activity and the dummy start activity of the next project. This virtual precedence relationship ensures the priority relations among projects cannot be broken when the crossover is carried out. Assume two individuals are selected for the crossover, which are

$$\pi^F = \left(A_{11}^F, \cdots, A_{1N_1}^F, \cdots, A_{L1}^F, \cdots, A_{LN_L}^F\right)$$

and

$$\pi^M = \left(A_{11}^M, \cdots, A_{1N_1}^M, \cdots, A_{L1}^M, \cdots, A_{LN_L}^M\right).$$

The process of crossover is as follows: First, a random integer $r(1 \leq r \leq \sum_{i=1}^{L} N_i)$ is generated as the position in the chromosome for crossover. Parental Chromosome π^F and π^M can produce their filial generation, a daughter and a son, through crossover at the position r. The first r genes of the son π^S directly come from the first r genes of π^F, and π^M provides the rest genes keeping their relative order unchanged.

$$\pi^S = (\underbrace{A_{11}^F, \cdots, A_{1N_1}^F, \cdots, A_{i1}^F, \cdots, A_{ij}^F}_{r}, A_{i(j+1)'}^M,$$

$$\cdots, A_{iN_i'}^M, \cdots, A_{L1}^M, \cdots, A_{LN_L}^M)$$

where

$$A_{ik'}^M \notin \left\{A_{11}^F, \cdots, A_{1N_1}^F, \cdots, A_{i1}^F, \cdots, A_{ij}^F\right\}, k = j+1, \cdots, N_i'.$$

The daughter can be generated with the same way,

$$\pi^D = (\underbrace{A_{11}^M, \cdots, A_{1N_1}^M, \cdots, A_{i1}^M, \cdots, A_{ij}^M}_{r}, A_{i(j+1)'}^F,$$

$$\cdots, A_{iN_i'}^F, \cdots, A_{L1}^F, \cdots, A_{LN_L}^F)$$

where

$$A_{ik'}^F \notin \left\{A_{11}^M, \cdots, A_{1N_1}^M, \cdots, A_{i1}^M, \cdots, A_{ij}^M\right\}, k = j+1, \cdots, N_i'.$$

Sönke Hartmann [25] has proved that the one-point crossover keeps the offspring precedence feasible as for the single project network encoding. The following theorem ensures that this property maintains for the encoding presented in section 4.1.

Theorem 1 *If adding the virtual precedence relationship to adjacent dummy activities in priority encoding, the*

precedence and priority relationships cannot be broken by the one-point crossover.

Proof. Because all the projects are catenated by the virtual precedence relationship like one project, where the priority encoding can be viewed as the encoding proposed by Sönke Hartmann [25]. Therefore, the priority encoding can also keep the offspring precedence relationship (including the virtual precedence relationship) unchanged. On the other hand, since the virtual precedence relationship is unchanged, the order of the projects in the chromosome is unchanged, that is, the priority relationship of the offspring maintains. □

4. Experiments

In this section, we will take experiments to test the effectiveness of NSGA-II based algorithm. Furthermore, it will focus on exploring the impacts of three parameters (OS, RC and UL) on the approximate Pareto optimal set gotten by NSGA-II based algorithm. Experimental instances and setups will be introduced in Section 4.1. The effectiveness of NSGA-II based algorithm is tested in Section 4.2. From Section 4.3 to 4.5, the analysis focus on three aspects:

(i) The impacts of UL on robustness. Obviously, the makespan has no relation with UL (refer to Section 4.3) .
(ii) The impacts of OS and RC on robustness and makespan, which are the objectives of the model constructed in Section 3 (refer to Section 4.4).
(iii) The relationship between the two objectives: robustness and makespan (refer to Section 4.5).

4.1. *Experiment setups*

In this section, we will describe the setup of experiments. In this field, several network generators for project scheduling problems have been developed[26-29]. The often-used instances in the project scheduling problem library PSPLIB have been generated using ProGen[26], which takes into account network topology and resource-related characteristics. In this paper, we use the RanGen software developed by Demeulemeester et al. [27, 28] to generate experimental instances. The reason is that it can generate strongly random instances that span the full range of problem complexity. Besides, it uses a non-superfluous reliable set of complexity measures, which have been used in former studies and shows its clear and strong relation to the hardness of resource-constrained project scheduling problems. It guarantees the network instance with pre-specified values of the number of activities, order strength (OS) and resource constrainedness (RC), which are also the main considered factors in this research.

The process of experiments be used in the following subsections is introduced briefly as follows.

Step 1: Set the number of activities to be 90, which belongs to three projects averagely. The priorities of the projects are descending from the first one to the last.
Step 2: From OS=0.1 to1 by 0.1
  From RC=0.1 to 1 by 0.1
    Step 2.1: Generate 100 multi-projects on the setting of OS and RC by RanGen;
    (Note, each multi-projects is composed of 90 activities belonging to three projects averagely.)
    Step 2.2: Denote the set of the 50 multi-projects as $P_{(OS,RC)}$;
    Step 2.3: Use NSGA-II based algorithm to calculate the schedules for each multi-project;
    Step 2.4: Denote $MS_{(OS,RC)}$ as the set of makespans of all multi-projects calculated by Eq.8;
    Step 2.5: For UL=0.1 to 0.9 by 0.2
      Calculate the robustness measure RM for the schedule of each instance under the uncertainty level represented by UL.
    End
  End
End

In the process, each set $P_{(OS,RC)}$ actually determines a class of instances. The priorities of the projects in an instance are descending from the first one to the last. Without loss of generality, set the resource constrainedness for all resource types with the same value, and RC_k is substituted by RC in the following. Note that the resource availability for each type is equal to that of the project composing the multi-project instance.

For the sake of efficiency, we set the population size and the number of generations to be 50 and 1000 respectively, to guarantee the algorithm can provide solutions of high quality within reasonable time. To offset the redundancy of decoding chromosomes, the crossover and mutation rate should be higher than the traditional set [24].

Therefore, we set the crossover and mutation rate to be 1.0 and 0.06.

4.2. *Effectiveness of NSGA-II based Algorithm*

According to the process of experiments, there are totally 10^4 instances according to different RC and OS. NSGA-II based algorithm is used to generate schedules for the instances. For the schedule of each instance, its robustness under different uncertainty levels is calculated. All the results show that NSGA-II based algorithm is effective and efficient. Fig.9 and Fig.10 show typical distributions of makespan and robustness under different settings of RC and OS respectively. In the subfigures of Fig.9 and Fig.10, X axis represents makespan (a) or RM (b) and Y axis represents the frequencies. Each histogram illustrates the frequency of instances with a makespan or RM under a setting of RC and OS. High values of frequency represent corresponding makespan or RM are more likely to obtain in the real schedule process. For the save of space, we only present the results in the case of UL=0.5 in two situations. For cases with small values of OS or RC, the distributions are quite similar with that of Fig.9. For other cases, most distributions are similar with Fig.10. The statistical distributions of makespan and robustness under different settings of RC and OS are listed in the appendix of this paper.

Fig.9 and Fig 10 show that our algorithm obtains solutions with large range of makespan and RM. This kind of distribution guarantees the diversity of the results, so that project managers can choose schedules according to their preference. In addition, stable schedules are successfully generated, especially when conflict is not strong (OS < 0.7 or RC < 0.7). The NSGA-II algorithm can also obtain relatively low value of makespan and RM even in strong conflict cases. In the next subsections, we will compute the average value of every frequency histogram to reflect the trend of variation of makespan and RM versus UL, OS and RC.

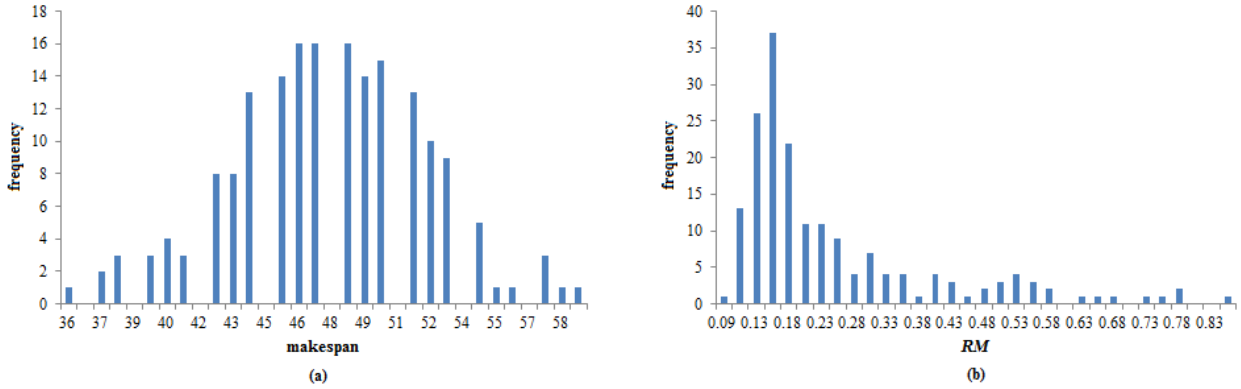


Fig.9 Distribution of makespan and RM when OS = 0.1 and RC = 0.1

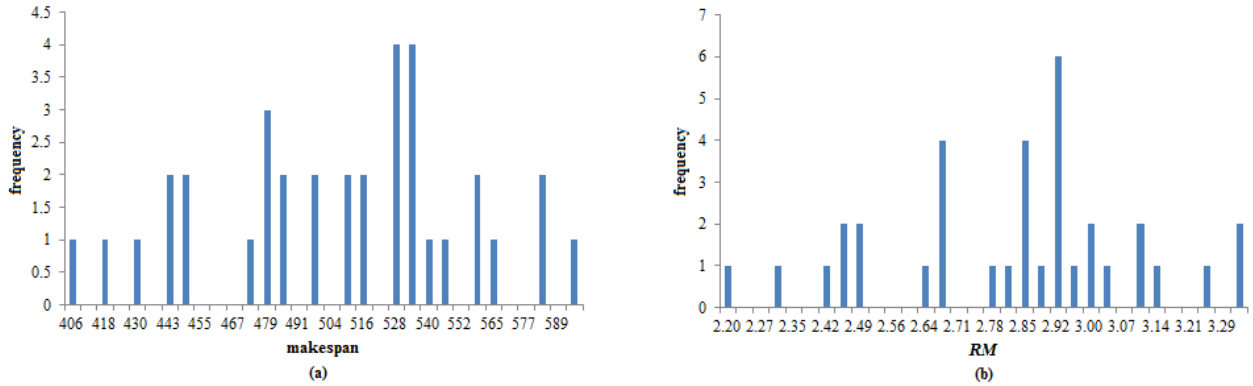


Fig.10 Distribution of makespan and RM when OS = 0.9 and RC = 0.9

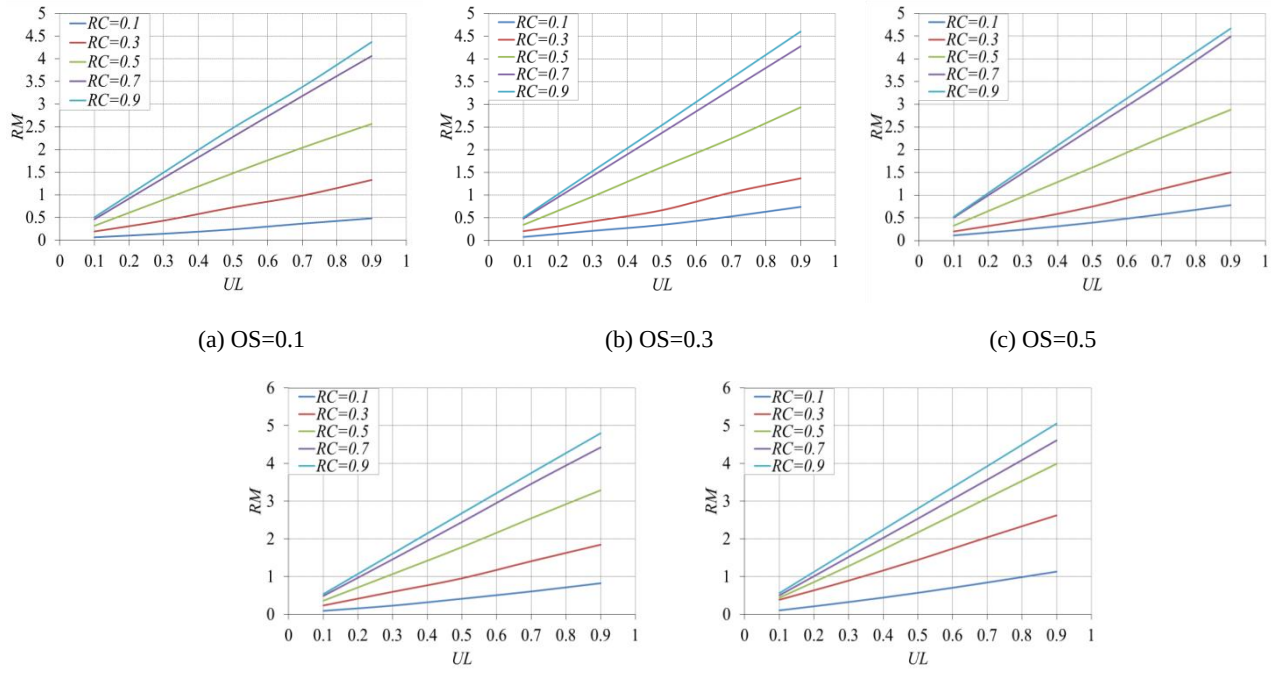
4.3. *Impacts of UL on robustness*

In this section, the impacts of UL on the robustness will be discussed. We compute the average value of RM for every frequency histogram in Fig.10 with different UL, OS and RC respectively. The average values represent the average RM of solutions belonging to the same class. Fig.11(a)-(e) show the relationship between the average value and parameters of UL, OS and RC. The results indicate that the UL almost linearly affects the average RM of a class for all combinations of OS and RC. The higher the UL the worse the robustness is. The feature offers us a rule to estimate the robustness performance of schedules under different ULs: when the UL increases, the RM increases almost linearly accordingly. Based on the rule, we can discuss the impact of OS and RC on solutions while ignoring the impact of UL. Therefore, it is fixed to 0.3 in Sections 5.2-5.4.

4.4. *Impacts of RC and OS on makespan*

Makespan is generally viewed as one of the most important issues considered in project scheduling. The stock charts in Fig.12 and Fig.13 show the relationship between makespan and parameters of RC and OS. For different combinations of RC and OS, although the makespan of a class of networks distributes in a range, the overall trend of the makespan (the dash lines) increases with RC and OS.

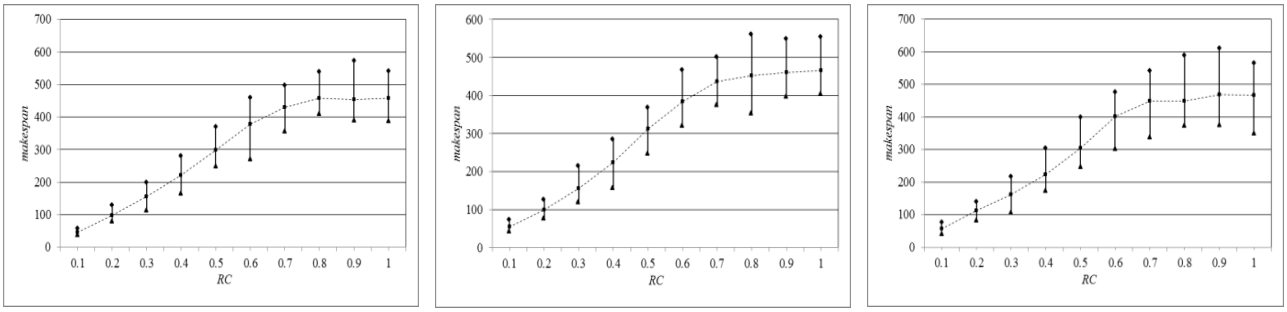
To get a full overview, we combine the trend lines of Fig.12 and Fig.13 into Fig.14. From Fig.14(a), we can see that the increasing rate of the makespan with RC varies in different intervals. The makespan increases faster in the case of $RC < 0.7$ than $RC > 0.7$. Especially, If $RC > 0.8$, the makespan is almost constant. The increasing rate of makespan versus RC is about 650 for all values of OS when RC is smaller than 0.7. Fig.14(b) clearly shows the relationship between the makespan and OS. The makespan increases with OS, and the increase rate is small. Especially, the increase rate is so small that it is close to zero when $OS \leq 0.7$. This means the OS plays a supporting role in affecting the makespan compared with RC.



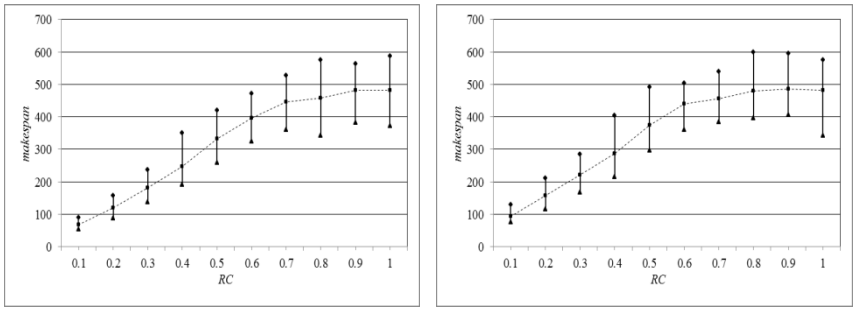
(a) OS=0.1 (b) OS=0.3 (c) OS=0.5

(d) OS=0.7 (e) OS=0.9

Fig.11 The impact of UL on the robustness

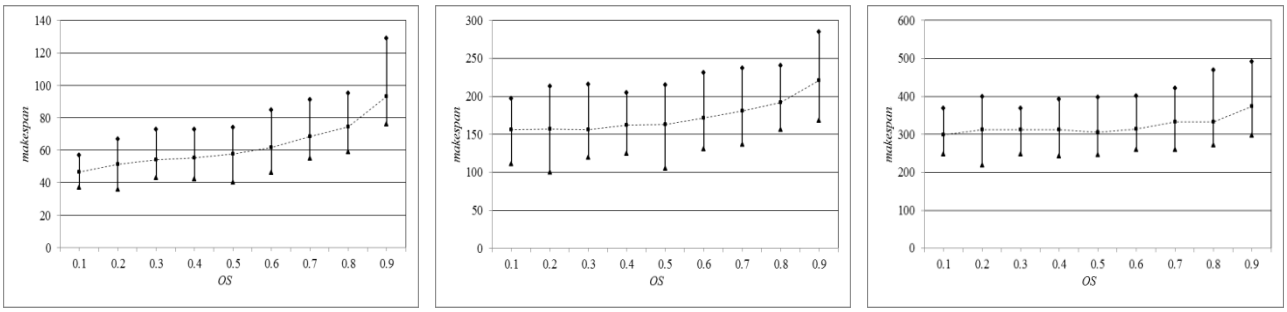


(a) OS=0.1 (b) OS =0.3 (b) OS =0.5

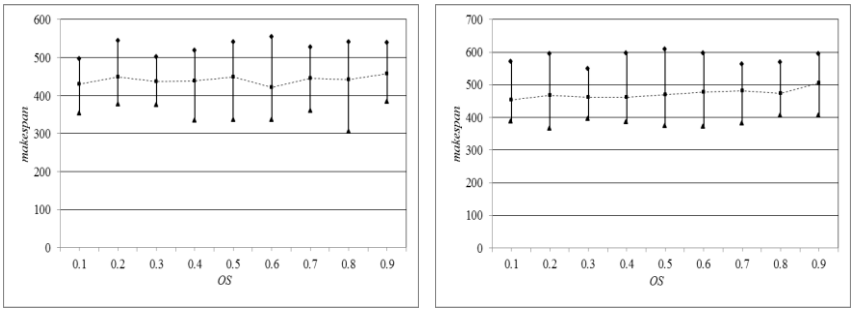


(d) OS =0.7 (e) OS =0.9

Fig.12 The impact of RC on the makespan

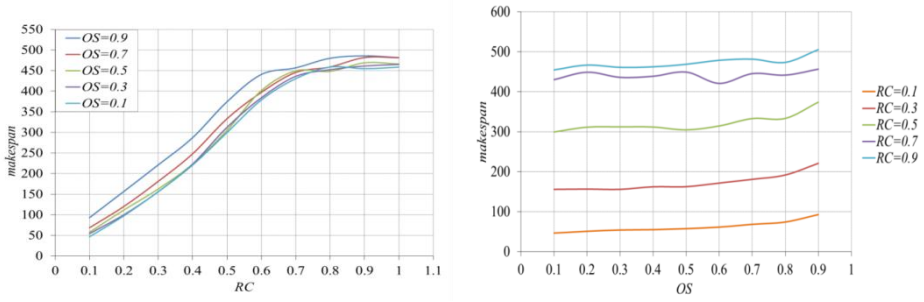


(a) RC=0.1 (b) RC =0.3 (b) RC =0.5



(d) RC =0.7 (e) RC =0.9

Fig.13 The impact of OS on the makespan



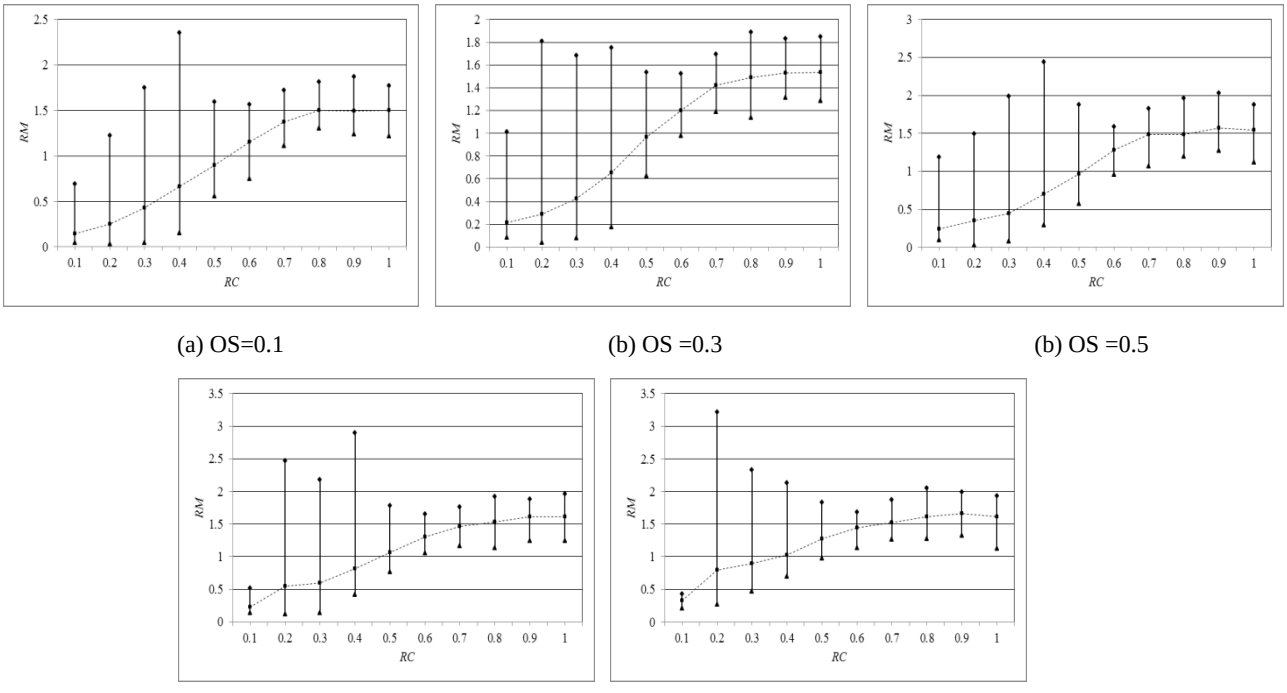
(a) (b)

Fig.14 The impacts of RC and OS on the makespan

4.5. *Impacts of RC and OS on the robustness*

The stock charts in Fig.15 and Fig.16 show the impacts of RC and OS on the robustness. In addition, we combine the trend lines into Fig.17. From the figures, we can conclude that: for different combinations of RC and OS, although the robustness of a class of networks distributes in a range, the overall trend of the robustness (the dash lines) increases with RC and OS. When the resource is ample or the complex of the network is low, the robustness of schedules will be high.

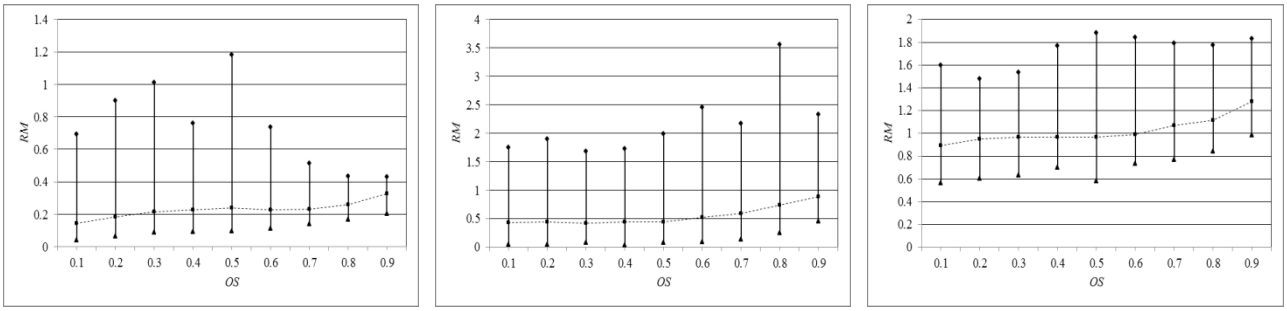
The results depicted in Fig.17(a) show that, RM is about twice of RC when $RC \leq 0.7$. It also can be seen from Fig.17(b) that RM changes slightly for all values of OS. That is to say, in general, there exist evident changes of RM as RC increases until RC reaches a threshold. Hence, it can be concluded that the robustness is remarkably impacted by the resource parameter while it is nearly indifferent with OS.



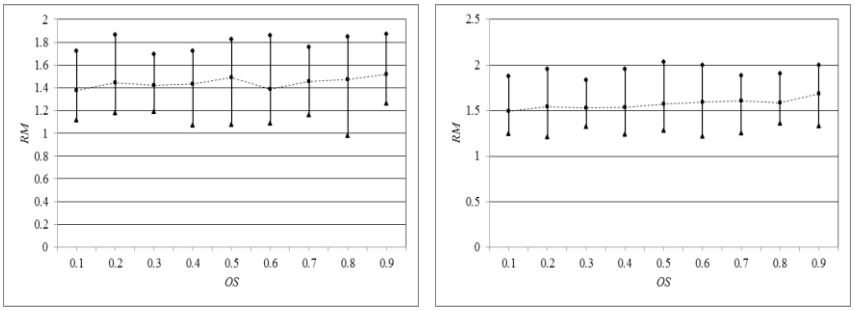
(a) OS=0.1 (b) OS =0.3 (b) OS =0.5

(d) OS =0.7 (e) OS =0.9

Fig.15 The impact of RC on the RM

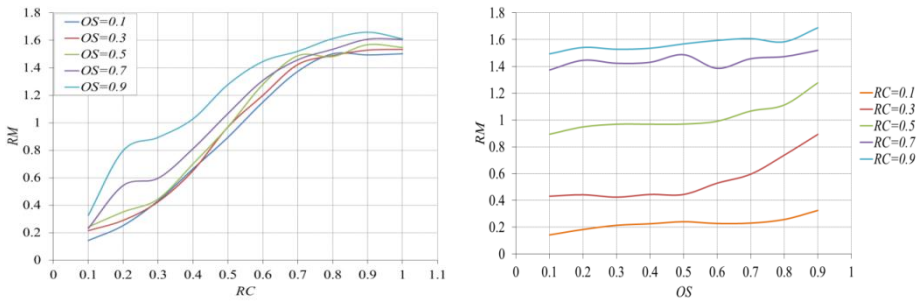


(a) RC=0.1 (b) RC =0.3 (b) RC =0.5



(d) RC =0.7 (e) RC =0.9

Fig.16 The impact of OS on the RM



(a) (b)

Fig.17 The impacts of RC and OS on the RM

4.6. *Relationship between robustness and makespan*

To explore the relationship between makespan and robustness, we first present the following notions to measure the increasing rate of robustness.

For the *j*th solution in the *i*th solution set belonging to a class of networks, the increasing rate of robustness is defined as:

$$\eta_{ij} = \frac{\mathrm{RM}_{ij} - \mathrm{RM}_{i(j+1)}}{\mathrm{makespan}_{i(j+1)} - \mathrm{makespan}_{ij}}$$

where RM_{ij} and $\mathrm{makespan}_{ij}$ are the values of RM and makespan of solution j in solution set i respectively. In this research, we suppose that the solutions in each solution set are sorted according to the makespan in ascending order (Note, it amounts to that RM ranks in descending order simultaneously). In this case, η_{ij} is always positive. If a solution set i contains only one element, set $\eta_{i1} = 0$.

Based on these, the average increasing rate of robustness for each class of instances is defined as:

$$\bar{\eta} = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i - 1}\sum_{j=1}^{n_i-1}\eta_{ij}$$

where m is the number of solution sets (instances) belonging to the same class, and n_i is the number of solutions of the solution set i.

$\bar{\eta}$ reflects the relationship between robustness and makespan of solutions. If $\bar{\eta}$ is greater than a threshold, the robustness fast increases as the makespan becomes longer and longer. Therefore, it is advantageous to choose the schedule with long makespan. If the robustness increases slowly, the schedule with long makespan may be not favorable. The threshold is related with the preference of decision makers. As a result, if the decision makers prefer a robust schedule, a lower threshold is suggested.

Fig.18 shows the impact of RC and OS on the increasing rate of robustness when $\mathrm{UL} = 0.3$. For each value of OS, the common characteristic of the curves is that the average increasing rate reaches the maximum point when RC is about 0.2. When RC is greater than 0.7, the average increasing rate is even close to zero. Obviously, RC has the same impact on the average increasing rate of robustness for different class of instances. No matter the resource is extremely sufficient or scarce, the robustness increases slowly with the increase of makespan. Especially, when the resource is scarce ($\mathrm{RC} > 0.7$), there exist little differences on robustness between solutions under the same set. That is to say, in this case schedules with relatively long makespan could hardly enhance robustness, compared with those with short makespan. The reason may lie in the fact that when resource is ample, all activities tend to be scheduled in parallel. When resource is scarce, they tend to execute serially. In both situations, the differences of makespan and robustness of schedules in the same solution set incline to disappear.

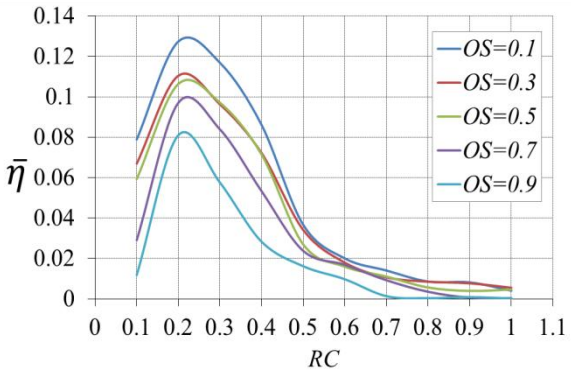


Fig.18 Impacts of RC and OS on $\bar{\eta}$

5. Results and Discussion

In general, the results of the experiments can be concluded as follows:

(i) Generally, projects with larger makespan have better robustness especially when resource is ample. Reasonable sacrifice of makespan usually offers more stable schedule. This improvement of robustness is remarkably when the level of uncertainty is high. Thus, a decision maker should choose an appropriate schedule instead of pursuing an absolutely shortest makespan according to different project environments.

(ii) As a key factor, the resource constrainedness affects both the makspan and robustness evidently. If the resource is ample, the makespan increases sharply as the amount of resource growing up. However, such impact becomes slight when the resource is relatively scarce. Similar laws can also be observed between the resource and the robustness. With the help of these observations, a decision maker can estimate the number of additional resource precisely based on the original amount of resources.

(iii) The impact of uncertainty on the robustness is approximately a linear relationship. In realistic dynamic project environment, the decision maker can estimate the performance of robustness by the uncertainty level to take measures upon uncertainty.

(iv) The effects of the order strength on the performance of scheduling are not as obvious as the uncertainty levels and resource constrainedness.

6. Conclusion

This paper mainly explores the modeling and solution of a new resource-constraint multi-project scheduling with priorities and uncertain activity durations. The robustness measure of the problem is another focus of this paper. Experiments are designed to testify the effectiveness of the proposed solution algorithm, and the impacts of uncertainty (UL), resource (RC) and network structure (OS) on the objectives of RCMPSPP are explored with experiments. Main contributions of this paper can be concluded as follows:

(i) A resource-constraint multi-project scheduling problems (RCMPSPP) with priority and uncertain activity durations is presented;

(ii) A bi-objective model for the problems and its solution algorithm are proposed;

(iii) A kind of robustness measure for RCMPSPP is presented and its related project parameters are given;

(iv) The impacts of the parameters on RCMPSPP are explored. The obtained experimental conclusions can be utilized as guidelines for practical applications.

In the future, we will extend our discussion on distributed multi-project scheduling problem and develop new solution algorithms to deal with it. Besides, the robustness analysis will also be extended to the distributed problem.

Reference

1. Payne, J.H., 1995. Management of multiple simultaneous projects: a state-of-the- art review. International Journal of Project Management, 13 (3), 163–168.
2. Kim K.W., Yun Y.S., Yoon J.M., Gen M., Yamazaki G., 2005. Hybrid genetic algorithm with adaptive abilities for resource-constrained multiple project scheduling. Computers in Industry, 56, 143–160.
3. Yassine A.A., Meier C., Browning T.R., 2007. Multi-Project Scheduling using Competent Genetic Algorithms, University of Illinois Department of Industrial & Enterprise Systems Engineering (IESE) Working Paper.
4. Goncalves J.F., Mendes J.J.M., Resende M.G.C., 2008. A genetic algorithm for the resource constrained multi-project scheduling problem, European Journal of Operational Research, 189, 1171–1190.
5. Lova, A., Tormos, P., 2001. Analysis of scheduling schemes and heuristic rules performance in resource-constrained multi-project scheduling. Annals of Operations Research, 102 (1–4), 263–286.
6. Lova, A., Tormos, P., 2002. Combining random sampling and backward-forward heuristics for resource-constrained multi-project scheduling. In: Proceedings of the Eight International Workshop on Project Management and Scheduling, Valencia, Spain, April, 244–248.
7. Lova, A., Maroto, C., Tormos, P., 2000. A multi-criteria heuristic method to improve resource allocation in multi-project scheduling. European Journal of Operational Research, 127, 408–424.
8. Browning T.R., Yassine A.A., 2010. Resource-constrained multi-project scheduling: Priority rule performance revisited, International Journal of Production Economics, 126, 212–228.
9. Mobini M., Mobini Z., Rabbani M., 2011. An artificial immune algorithm for the project scheduling problem under resource constraints, Applied Soft Computing Journal, 11(2), 1975-1982.
10. Ziarati K., Akbari R., Zeighami V., 2011. On the performance of bee algorithms for resource-constrained project scheduling problem, Applied Soft Computing Journal, 11(4), 3720-3733.
11. Caniëls M.C.J. Bakens R.J.J.M, 2012. The effects of Project Management Information Systems on decision making in a multi project environment, 30, 162-175.
12. Scholl D.K.A., 2009. A heuristic solution framework for the resource constrained (multi-)project scheduling problem with sequence-dependent transfer times, European Journal of Operational Research, 197, 492–508.
13. Leus R., 2003. The generation of stable project plans. PhD thesis, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.
14. Hansa E.W., Herroelenb W., Leusb R., Wullink G., 2007. A hierarchical approach to multi-project planning under uncertainty, Omega, 35, 563 – 577.
15. Fox B.R., Ringer M., 1995. Planning and scheduling benchmarks, Homepage (March 6, 1995) posted at ftp://ftp.neosoft.com/pub/users/b/benchmrx/homepage.html .
16. Herroelen W., Leus R., 2005. Project scheduling under uncertainty: Survey and research potentials, European Journal of Operational Research, 165, 289–306.
17. Van de Vonder S., Demeulemeester E., Herroelen W., Leus R., 2005. The use of buffers in project management: The trade-off between stability and makespan, International Journal of Production Economics, 97, 227–240.
18. Van de Vonder S., Demeulemeester E., Herroelen W., 2008. Proactive heuristic procedures for robust project scheduling: An experimental analysis, European Journal of Operational Research, 189, 723–733.
19. Lambrechts O., Demeulemeester E., Herroelen W., 2007. A tabu search procedure for developing robust predictive project schedules. International Journal of Production Economics, 111 (2), 496–508.
20. Lambrechts O., Demeulemeester E., Herroelen W., 2008. Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. Journal of Scheduling, 11 (2), 121–36.
21. Al-Fawzana M.A., Haouari M., 2005. A bi-objective model for robust resource-constrained project scheduling. International Journal of Production Economics, 96, 175–187.
22. Herroelen W., Leus R., 2004. The construction of stable project baseline schedules. European Journal of Operational Research, 156, 550–565.
23. Mastor A.A., 1970. An experimental investigation and comparative evaluation of production line balancing techniques. Management Science, 16(11), 728-746.
24. Deb K., Pratap A., Agarwal S., Meyarivan T., 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. On Evolutionary Computation, 6 (2), 182–197.
25. Hartmann S., 1998. A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling, Naval Research Logistics, 45, 733–750.
26. Kolisch R., Sprecher A., Drexl A., 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science, 41, 1693–1703.
27. Demeulemeester E., Vanhoucke M., Herroelen W., 2003. RanGen: A random network generator for activity-on-the-node networks. Journal of Scheduling, 6, 17–38.
28. Demeulemeester E., Dodin B., Herroelen W., 1993. A random activity network generator. Oper. Res., 41, 972–980.
29. Agrawal M.K., Elmaghraby S.E., Herroelen W., 1996. DAGEN: a generator of testsets for project activity nets. European Journal of Operational Research, 90, 376–382.

Appendix A. Distribution of makespan and RM

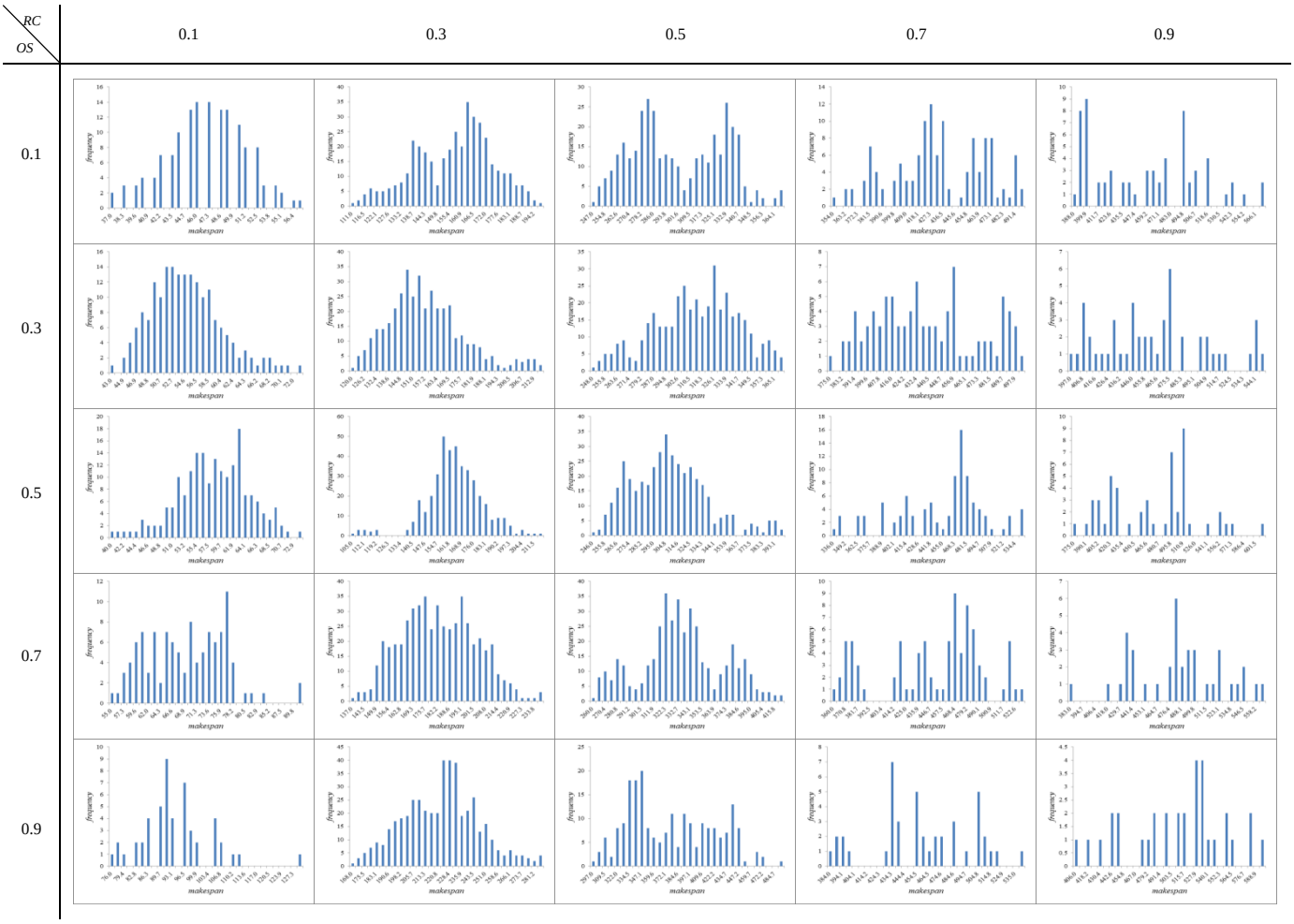


Fig.19 Distribution of makespan under different settings of RC and OS

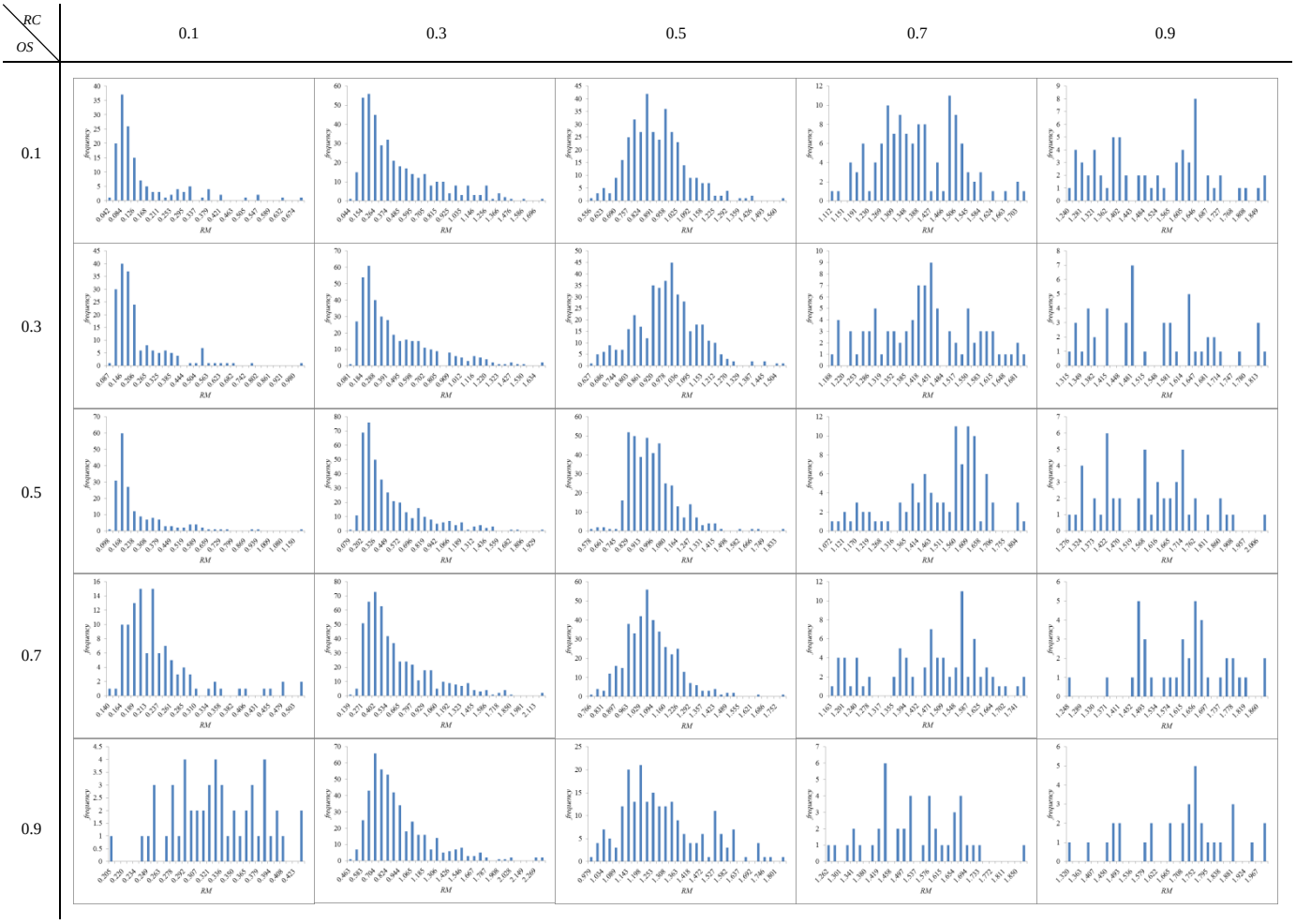


Fig.20 Distribution of RM under different settings of RC and OS