Chen WN, Zhang J. Scheduling multi-mode projects under uncertainty to optimize cash flows: A Monte Carlo ant colony system approach. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 27(5): 950–965 Sept. 2012. DOI 10.1007/s11390-012-1276-2

Scheduling Multi-Mode Projects under Uncertainty to Optimize Cash Flows: A Monte Carlo Ant Colony System Approach

Wei-Neng Chen (陈伟能), *Member IEEE*, and Jun Zhang (张　军), *Senior Member*, *IEEE*

Department of Computer Science, Sun Yat-sen University, Guangzhou 510006, China

E-mail: chenwn3@mail.sysu.edu.cn; junzhang@ieee.org

Received September 3, 2011; revised July 13, 2012.

Abstract Project scheduling under uncertainty is a challenging field of research that has attracted increasing attention. While most existing studies only consider the single-mode project scheduling problem under uncertainty, this paper aims to deal with a more realistic model called the stochastic multi-mode resource constrained project scheduling problem with discounted cash flows (S-MRCPSPDCF). In the model, activity durations and costs are given by random variables. The objective is to find an optimal baseline schedule so that the expected net present value (NPV) of cash flows is maximized. To solve the problem, an ant colony system (ACS) based approach is designed. The algorithm dispatches a group of ants to build baseline schedules iteratively using pheromones and an expected discounted cost (EDC) heuristic. Since it is impossible to evaluate the expected NPV directly due to the presence of random variables, the algorithm adopts the Monte Carlo (MC) simulation technique. As the ACS algorithm only uses the best-so-far solution to update pheromone values, it is found that a rough simulation with a small number of random scenarios is enough for evaluation. Thus the computational cost is reduced. Experimental results on 33 instances demonstrate the effectiveness of the proposed model and the ACS approach.

Keywords project scheduling, optimization under uncertainty, cash flow, ant colony optimization, Monte Carlo simulation

1 Introduction

Project scheduling is one of the most important subjects in the project management field. The classical project scheduling model, namely the resource constrained project scheduling problem (RCPSP), involves scheduling the activities of a project to minimize makespan subject to precedence and resource constraints. The problem has been proven to be NP-hard[1-2]. A more general formulation is the multi-mode RCPSP (MRCPSP)[3-4]. It considers a common situation in real-life projects that activities can be performed by different alternative modes. These modes consume different quantities of time, cost, and resources to complete the same activity. To handle the time/cost/resource trade-offs among these modes, MRCPSP has to deal with not only the processing order of activities but also the selection of execution modes. As a result, the computational complexity of MRCPSP significantly increases. It has been demonstrated by Kolisch[5] that finding a feasible solution for the MRCPSP with more than one nonrenewable resource is already NP-complete, let alone the problem of finding the optimal feasible solution. While the conventional project scheduling models use makespan as the criterion, financial criteria have been increasingly considered to be more meaningful for capital-intensive projects [6-7]. The most commonly used financial criterion is the net present value (NPV) of discounted cash flows. NPV is defined as the difference between cash inflows and outflows, taking into account the time value of money by discounting the cash flows. The presence of the NPV criterion results in a more complicated model called MRCPSP with discounted cash flows (MRCPSPDCF)[8-10].

Due to the importance and difficulty of project scheduling, a considerable amount of research effort has been devoted to proposing various project scheduling algorithms during the last decades. However, in most studies, the scheduling models are based on a premise that the environmental parameters such as activity durations and costs can be determined before scheduling. In application, due to the occurrence of unexpected situations, it is almost impossible to determine

Regular Paper

This work was supported in part by the National Science Fund for Distinguished Young Scholars of China under Grant No. 61125205, the National Natural Science Foundation of China (NSFC) under Grant No. 61070004, NSFC-Guangdong Joint Fund under Key Project No. U0835002.

©2012 Springer Science + Business Media, LLC & Science Press, China

all environmental parameters of a project a priori[11-13]. As the actual durations may be somewhat longer or shorter than expected, the probability of processing a project exactly following the predefined plan is very low. As a result, the disrupted schedule may miss deadlines and incur deficit, and the validity of the schedules generated by deterministic scheduling has been heavily questioned[11]. For the scheduling models with financial aspects, forecasting the cash inflows and outflows accurately is a very difficult task[14]. In this sense, handling uncertainty is even more important in the model with financial aspects. Generally speaking, to be a practical scheduling model for project selection and planning, uncertainty is an innegligible factor.

Scheduling under uncertainty is a challenging field of research that has attracted increasing attention. Several types of scheduling models and algorithms have been proposed for the classical single-mode RCPSP under uncertainty with the makespan criterion[15-20]. However, for the scheduling problem with financial aspects, the literature that considers uncertainty is rather sparse[11]. In addition, the existing studies[21-25] only consider the single-mode scheduling problem and disregard the resource constraints in the project. To the best of our knowledge, the more realistic and complicated model with resource constraints and alternative execution modes is still not considered in the literature.

This paper intends to deal with a multi-mode resource constrained project scheduling problem under uncertainty with the objective to maximize the NPV of cash flows. We term the scheduling model the stochastic MRCPSPDCF (S-MRCPSPDCF). In the model, uncertainty is sourced from activity durations and costs. The duration and cost of every alternative execution mode are given by random variables of certain probability distributions. Due to the presence of random environmental parameters, the scheduling problem can be divided into two stages: the pre-execution stage before knowing the actual values of random variables and the post-execution stage when all random variables have been revealed. In real-world application, in order to evaluate the potential of a project, it is necessary to first build an efficient baseline schedule in the pre-execution stage. With this requirement, the objective of the considered S-MRCPSPDCF is actually to find a baseline schedule in the pre-execution state, so that for all possible realization of the random variables in the post-execution state, the expected NPV of cash flows is maximized.

To solve this problem, this paper proposes an ant colony optimization (ACO) approach. ACO is a population-based optimization algorithm proposed by Dorigo[26-27] in the early 1990s in the light of how ants manage to discover the shortest path from their nest to a food source. It has been successfully applied to many complicated scheduling problems[28-30]. Recently, Gutjahr[31-32] and Balaprakash[33] have developed a simulation ACO algorithm (S-ACO) to tackle the probabilistic traveling salesman problem (PTSP), demonstrating the great potential of ACO for solving optimization problems under uncertainty. Inspired by these studies, this paper develops an ant colony system (ACS) approach to S-MRCPSPDCF. In the algorithm, artificial ants build baseline schedules iteratively according to pheromones and an expected discounted cost (EDC) heuristic. The EDC heuristic is defined based on the expected values of activity durations and costs. Since the model has uncertainties, it is impossible to evaluate the expected NPV of a baseline schedule through a deterministic expression. In this situation, the proposed algorithm adopts the Monte Carlo (MC) simulation. Thus the algorithm is termed MC-ACS. MC simulation is a computational method that computes the properties of a system by repeatedly sampling from the system using random numbers. If we need very accurate results, we usually have to sample a large number of simulation scenarios, and thus the computational cost of the simulation is very high. Fortunately, different from other ACO variants, ACS only uses the best-so-far solution to update pheromones. Therefore, the proposed algorithm only needs a rough simulation with a small number of simulation scenarios to distinguish the best-so-far solution. In addition, a special technique that increases the sample size linearly[31] is applied. These design schemes have the function of reducing the computational cost for the algorithm. The proposed algorithm is tested on 33 randomly-generated instances. Experimental results show that the proposed approach is effective.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work on RCPSP and scheduling under uncertainty. Section 3 formulates the considered S-MRCPSPDCF model. Section 4 presents the proposed ACO approach. Experimental studies are shown in Section 5, and the conclusions are finally summarized in Section 6.

2 Background and Related Work

2.1 Resource Constrained Project Scheduling

Resource constrained project scheduling is an important model that has broad applications in industrial projects. Given a project with n activities, the goal of the conventional RCPSP is to arrange the processing

time of each activity subject to the precedence and resource constraints so that the makespan of the project is minimized. To find a feasible schedule, the most common approach is to find an activity list that specifies an order of the activities satisfying the precedence relations[1]. Then the activities are assigned to feasible processing time according to their order in the activity list. Various methods have been proposed to find the activity list, for example, exact approaches like the branch-and-bound algorithm[34] and meta-heuristic approaches like tabu search (TS), simulated annealing (SA), genetic algorithm (GA)[35], and ACO[28].

In conventional RCPSP, each activity can only be implemented in a single mode. In practice, it is common that an activity can be implemented in multiple ways. For example, an activity can be done by 10 employees and four machines in four months in the normal mode. We can also devote 20 employees and six machines to the activity to shorten its duration to three months in the urgent mode. Therefore, the different execution modes represent different trade-offs among time, cost and resource for an activity. The more general form of RCPSP that considers multiple execution modes is called multi-mode RCPSP. To handle the trade-offs in the MRCPSP, we have to determine not only the activity list, but also the mapping of each activity to a suitable execution mode. Thus the computational complexity of MRCPSP is significantly increased. By now, the existing approaches to MRCPSP are mainly meta-heuristics, for example, hybrid GA[3-4], ACO[36], SA[37]. For a more comprehensive survey to MRCPSP, please refer to [1].

The traditional RCPSP and MRCPSP only consider the makespan criterion and ignore the financial aspect. In recent years, financial aspect is increasingly considered to be crucial to project scheduling. The most commonly used financial criterion is NPV[6]. Compared with the original makespan criterion, the NPV criterion incurs higher computational cost for an evaluation of the objective function, as the objective function is nonlinear. In addition, as the evaluation of NPV involves both the time and cost of a project, it is more difficult to define effective heuristics for the problem. Due to the difficulty and complexity of MRCPSPDCF, by now, only several meta-heuristic approaches have been developed for the problem, for example, the GA approach proposed by Ulusoy[8], the SA and TS methods proposed by Mika *et al.*[9], and the ACO approach proposed by Chen *et al.*[10].

In this paper, we extend our previous work on MRCPSPDCF[10] to make the model and the ACO approach more practical by considering uncertainties. Different from the previous study, in order to address uncertainty, we incorporate MC simulation in the ACO approach and develop an expected discounted cost (EDC) based heuristic for the problem.

2.2 Scheduling under Uncertainty

Scheduling under uncertainty is a new and challenging direction in operations research. According to Bianchi *et al.*[38-39] and Jin and Branke[40], the existing approaches for handling uncertainty in scheduling can be generally classified into four types: reactive models, stochastic models, fuzzy models, and robust models. Reactive scheduling models[41] arise when environmental parameters are highly uncertain. In this case, it is impossible to build a solution that is usable for any realization of the environmental parameters, and thus activities are scheduled on-line with the execution of the project. Stochastic models are by now the most commonly used models for modeling uncertainty[15-18]. The main feature of stochastic models is that parameters with uncertainty are described by random variables. Under this assumption, the objective function in a stochastic scheduling model strongly depends on the probabilistic structure of the model. Some typical objective functions for this kind of scheduling models involve minimizing the expected makespan, minimizing the expected cost, and maximizing the probability of obeying all constraints of the project. Different from stochastic models, in the fuzzy models[19] and robust models[20], uncertainty is modeled by fuzzy quantities and interval values, respectively. For a comprehensive survey of scheduling under uncertainty, please refer to [11-12].

Although various models for scheduling under uncertainty have been developed, the above-mentioned models only consider single-mode projects with uncertainties only on activity durations. As it is difficult to accurately forecast the cash flows of a project, handling uncertainty is even more important in the project scheduling models with the NPV criterion[13]. However, existing studies that consider uncertainty in project scheduling with financial aspects are rather sparse[11]. The studies of cash flow optimization in project scheduling under uncertainty only include the dynamic scheduling policy based on the backward stochastic dynamic programming recursion method[21-22], the dynamic scheduling algorithm[23], and the branch-and-bound approaches[24-25]. In addition, in all these approaches, the scheduling models are somewhat simplified. For one thing, the resource constraints are either disregarded[21-24] or relaxed to the single-machine constraint[25]. For another, only a single execution mode is considered for each activity. Overall, it is desirable to further extend existing models and approaches to consider resource constraints, multiple execution modes and uncertainty in financial aspects.

3 Formulation of Stochastic MRCPSPDCF

3.1 Problem Description

S-MRCPSPDCF is a problem of scheduling a multi-mode project with uncertain durations and costs, subject to precedence and resource constraints, with the objective to maximize the expected NPV. In the model, a project is described by the following features.

3.1.1 Precedence Constraint

In the model, a project is described by an activity-on-arc (AoA) network $G = (E, A)$. The node set $E = \{e_1, e_2, \ldots, e_{|E|}\}$ corresponds to the set of events in the project, where $|E|$ is the number of events. The arc set $A = \{a_1, a_2, \ldots, a_n\}$ corresponds to the set of activities, where n is the number of activities. The AoA network defines the precedence relations between the activities. An example of the AoA network is shown in Fig.1.

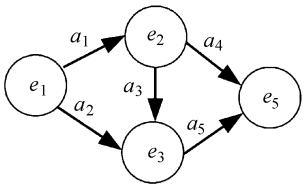

Fig.1. Example of the AoA network.

3.1.2 Resource Constraint

Renewable resource constraints are considered in the model. Suppose the project uses R types of renewable resources, for each time, the consumption of the k-th ($k = 1, 2, \ldots, R$) renewable resource is limited to R^k units.

3.1.3 Multi-Mode

The model considered in this paper allows each activity to be implemented in different ways. Each activity a_i ($i = 1, 2, \ldots, n$) is associated with a set of execution modes $M_i = \{m_{i1}, m_{i2}, \ldots, m_{i|M_i|}\}$, where m_{ij} is the j-th mode for the execution of a_i and $|M_i|$ is the total number of available modes for a_i. To complete an activity a_i, the different modes corresponding to a_i may consume different quantities of time, cost and resources. We denote the duration and cost of mode m_{ij} as d_{ij} and c_{ij}, respectively. Mode m_{ij} also consumes r_{ij}^k units of the k-th renewable resource. In the model, all of the execution modes are non-preemptive. That is, once a mode m_{ij} begins, it has to be complete without any interruption.

3.1.4 Uncertain Durations and Costs

Different from the traditional deterministic scheduling models, S-MRCPSPDCF takes account of uncertainty. In application, a common approach for predicting durations and costs is to build frequency distributions based on historical data[42]. Therefore, in this paper, uncertainty is formulated as a stochastic scheduling model.

In the model, duration d_{ij} and cost c_{ij} of mode m_{ij} are given by random variables instead of deterministic numbers. The random variables are of some pre-given probability distributions (e.g., normal and beta distributions). For the sake of convenience, we suppose that the durations are positive integer valued random variables and the costs are non-negative random variables in the form of either integer or real numbers. In addition, lower and upper bounds are defined to control the domain of durations and costs. The durations and costs of all execution modes are represented by a random matrix $\boldsymbol{\omega}$.

$$\boldsymbol{\omega} = \begin{pmatrix} d_{11}, c_{11}, d_{12}, c_{12}, & \cdots, & d_{1|M_1|}, c_{1|M_1|} \\ d_{21}, c_{21}, d_{22}, c_{22}, & \cdots, & d_{2|M_2|}, c_{2|M_2|} \\ & \vdots & \\ d_{n1}, c_{n1}, d_{n2}, c_{n2}, & \cdots, & d_{n|M_n|}, c_{n|M_n|} \end{pmatrix}. \quad (1)$$

Due to the presence of random variables, the scheduling problem can be divided into two stages. At the pre-execution stage, we need to first build an efficient baseline schedule. Then at the post-execution stage, according to the realization of the random variables, the baseline schedule is converted into an actual schedule, and the makespan and NPV of the actual schedule can be computed. The goal of the considered S-MRCPSPDCF is to find an optimal baseline schedule that maximizes the expected NPV under all possible realizations.

3.2 From Baseline Schedules to Actual Schedules

Due to the presence of random variables, the NPV of a baseline schedule cannot be evaluated directly. To evaluate the NPV, we need to first convert the baseline schedule into an actual schedule under a certain realization of the random variables. In this subsection, we discuss how to convert a baseline schedule to an actual schedule under a certain realization. A baseline schedule $\boldsymbol{S}$ for S-MRCPSPDCF is in the form of

$$\boldsymbol{S} = \begin{pmatrix} (a_{p1}, a_{p2}, \ldots, a_{pn}) \\ (k_{p1}, k_{p2}, \ldots, k_{pn}) \end{pmatrix}, \quad (2)$$

where $p_1, p_2, \ldots, p_n$ is a permutation of the numbers from 1 to n, and k_{p_i} means that activity a_{p_i} is mapped to execution mode $m_{p_i k_{p_i}}$. Here $\boldsymbol{S}$ actually defines the processing order $(a_{p_1}, a_{p_2}, \ldots, a_{p_n})$ of activities and

the selection $(k_{p_1}, k_{p_2}, \ldots, k_{p_n})$ of execution modes. To guarantee the satisfaction of precedence constraints, if a_i is a predecessor activity of a_j in the AoA network, a_i must appear earlier than a_j in the processing order $(a_{p_1}, a_{p_2}, \ldots, a_{p_n})$.

We suppose the realization of the random matrix $\boldsymbol{\omega}$ is $\boldsymbol{\omega}'$.

$$\boldsymbol{\omega}' = \begin{pmatrix} d'_{11}, c'_{11}, d'_{12}, c'_{12}, & \cdots, & d'_{1|M_1|}, c'_{1|M_1|} \\ d'_{21}, c'_{21}, d'_{22}, c'_{22}, & \cdots, & d'_{2|M_2|}, c'_{2|M_2|} \\ & \vdots & \\ d'_{n1}, c'_{n1}, d'_{n2}, c'_{n2}, & \cdots, & d'_{n|M_n|}, c'_{n|M_n|} \end{pmatrix}, \tag{3}$$

where d'_{ij} and c'_{ij} are the realizations of d_{ij} and c_{ij}, respectively. Under $\boldsymbol{\omega}'$, the baseline schedule $\boldsymbol{S}$ can be converted into an actual schedule $\boldsymbol{S}(\boldsymbol{\omega}')$ using the serial schedule generation scheme (SSGS)[43]. The pseudo-code of the SSGS is shown in Fig.2. The basic idea of SSGS is to repeatedly pick the first unselected activity from the processing order $(a_{p_1}, a_{p_2}, \ldots, a_{p_n})$ of activities, map it to its corresponding execution mode, and schedule it to the earliest possible time that satisfies the precedence and resource constraints to execute. An example of converting $\boldsymbol{S}$ to actual schedules under different realizations is illustrated in Fig.3. Given a baseline schedule $\boldsymbol{S}$ with the processing order $(a_2, a_1, a_4, a_3, a_5)$ and the selection of modes (1, 2, 2, 3, 1), we first select the first activity a_2 from the order and map a_2 to its first mode m_{21}. Then m_{21} is scheduled to the beginning time of the project. Then the next activity a_1 is picked and scheduled. These procedures run repeatedly until all activities have been arranged. According to Fig.3, it can be seen that a baseline schedule can be converted into different actual schedules under different realizations of the random variables.

Procedure SSGS($\boldsymbol{S}, \boldsymbol{\omega}'$)
1 Initialize; /*All units of resources are set available
2 **for** $i = 1$ to $n + 1$
3 Select the activity a_{p_i} from the list $(a_{p_1}, a_{p_2}, \ldots, a_{p_n})$ in $\boldsymbol{S}$;
4 Map a_{p_i} to the execution mode $m_{p_i k_{p_i}}$;
5 Schedule $m_{p_i k_{p_j}}$ to the earliest possible start time t to execute subject to precedence and resource constraints;
6 $\boldsymbol{S}(\boldsymbol{\omega}').ST_{ik_i} \leftarrow t$;
7 Update resources;
8 **end for**
9 Set $\boldsymbol{S}(\boldsymbol{\omega}').FT$ to the time when all the execution modes have finished;
end procedure

Fig.2. Pseudo-code of SSGS.

After converting the baseline schedule $\boldsymbol{S}$ into an actual schedule $\boldsymbol{S}(\boldsymbol{\omega}')$ under the realization $\boldsymbol{\omega}'$, the start time and end time of each execution mode in the actual schedule $\boldsymbol{S}(\boldsymbol{\omega}')$ can be determined. Here if activity a_i is executed in mode m_{ik_i} in $\boldsymbol{S}$, we denote the start time of m_{ik_i} in $\boldsymbol{S}(\boldsymbol{\omega}')$ as $\boldsymbol{S}(\boldsymbol{\omega}').ST_{ik_i}$ and the finish time of $\boldsymbol{S}(\boldsymbol{\omega}')$ as $\boldsymbol{S}(\boldsymbol{\omega}').FT$.

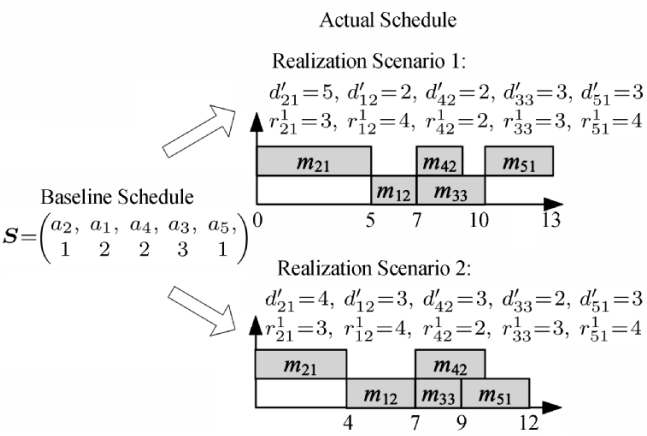

Fig.3. Example of converting a baseline schedule into different actual schedules under different realizations of random variables. In the example, the AoA network of the project is given in Fig.1. The resource constraint is set to $R^1 = 6$.

3.3 Evaluation of NPV of Cash Flows

Once the execution time of all execution modes have been determined, we can further calculate the actual NPV of cash flows $NPV(\boldsymbol{S}(\boldsymbol{\omega}'))$ of the schedule $\boldsymbol{S}(\boldsymbol{\omega}')$ by

$$NPV(\boldsymbol{S}(\boldsymbol{\omega}')) = Inflow(\boldsymbol{S}(\boldsymbol{\omega}')) - Outflow(\boldsymbol{S}(\boldsymbol{\omega}')) + BP(\boldsymbol{S}(\boldsymbol{\omega}')), \tag{4}$$

where $Inflow(\boldsymbol{S}(\boldsymbol{\omega}'))$, $Outflow(\boldsymbol{S}(\boldsymbol{\omega}'))$ and $BP(\boldsymbol{S}(\boldsymbol{\omega}'))$ represent the cash inflows, cash outflows, and bonus or penalty, respectively.

3.3.1 Cash Inflows

Cash inflows occur when the contractor pays for the execution of the project. At the beginning of the project, suppose U is the total project value and λ is the prepayment rate, the startup cash inflows $SInflow(\boldsymbol{S}(\boldsymbol{\omega}'))$ is given by

$$SInflow(\boldsymbol{S}(\boldsymbol{\omega}')) = \lambda U. \tag{5}$$

During the course of the project, the contractor pays for the finishing activities at some predefined milestone events. The cash flows are discounted according to a discounted rate α. If W_i is the net worth of a_i, θ is the milestone payment rate, and the payment for a_i under $\boldsymbol{S}(\boldsymbol{\omega}')$ occurs at time $\boldsymbol{S}(\boldsymbol{\omega}').PT_i$, then the milestone

inflows $MInflow(\boldsymbol{S}(\boldsymbol{\omega}'))$ is given by

$$MInflow(\boldsymbol{S}(\boldsymbol{\omega}')) = \sum_{i=1}^{n} \theta W_i \times \exp(-\alpha \times \boldsymbol{S}(\boldsymbol{\omega}').PT_i). \quad (6)$$

Finally, the contractor pays the rest unpaid values for the project, which is given by

$$FInflow(\boldsymbol{S}(\boldsymbol{\omega}')) = \Big(U - \lambda U - \sum_{i=1}^{n} \theta W_i\Big) \times \exp(-\alpha \times \boldsymbol{S}(\boldsymbol{\omega}').FT). \quad (7)$$

So the present value of cash inflows can be computed as follows.

$$Inflow(\boldsymbol{S}(\boldsymbol{\omega}')) = SInflow(\boldsymbol{S}(\boldsymbol{\omega}')) + MInflow(\boldsymbol{S}(\boldsymbol{\omega}')) + FInflow(\boldsymbol{S}(\boldsymbol{\omega}')). \quad (8)$$

3.3.2 Cash Outflows

Cash outflows involve the cost of executing the project. If activity a_i is executed in mode m_{ik_i} in schedule $\boldsymbol{S}$, under realization $\boldsymbol{\omega}'$, the actual duration and cost of m_{ik_i} are d'_{ik_i} and c'_{ik_i}, respectively. So the discounted cash outflows can be evaluated as follows.

$$Outflow(\boldsymbol{S}(\boldsymbol{\omega}')) = \sum_{i=1}^{n} c'_{ik_i} \times \exp(-\alpha \times (\boldsymbol{S}(\boldsymbol{\omega}').ST_{ik_i} + d'_{ik_i})). \quad (9)$$

3.3.3 Bonus or Penalty

The cash flows of bonus (or penalty) arise when the project finishes earlier (or later) than a predetermined interval of due dates $[T_{\mathrm{LOW}}, T_{\mathrm{UP}}]$. If the finish time $\boldsymbol{S}(\boldsymbol{\omega}').FT$ is earlier than T_{LOW}, the company will gain a bonus payment. Otherwise, if $\boldsymbol{S}(\boldsymbol{\omega}')$ comes later than T_{UP}, the company will lose money. Suppose γ and δ are the bonus and penalty rates, the cash flows of bonus or penalty are given by

$$BP(\boldsymbol{S}(\boldsymbol{\omega}')) = \begin{cases} \gamma U(T_{\mathrm{LOW}} - \boldsymbol{S}(\boldsymbol{\omega}').FT) \times \\ \quad \exp(-\alpha \times \boldsymbol{S}(\boldsymbol{\omega}').FT), \\ \qquad \text{if } \boldsymbol{S}(\boldsymbol{\omega}').FT < T_{\mathrm{LOW}}, \\ 0, \quad \text{if } T_{\mathrm{LOW}} \leqslant \boldsymbol{SS}(\boldsymbol{\omega}').FT \leqslant T_{\mathrm{UP}} \\ -\delta U(\boldsymbol{S}(\boldsymbol{\omega}').FT - T_{\mathrm{UP}}) \times \\ \quad \exp(-\alpha \times \boldsymbol{S}(\boldsymbol{\omega}').FT), \\ \qquad \text{if } T_{\mathrm{UP}} < \boldsymbol{S}(\boldsymbol{\omega}').FT. \end{cases} \quad (10)$$

Since a baseline schedule is not an actual schedule for execution, to evaluate the performance of a baseline schedule, the S-MRCPSPDCF model adopts the expected NPV of cash flows under all possible realizations as the criterion. If the random variables in $\boldsymbol{\omega}$ are of discrete values, the NPV expectation of a baseline schedule $\boldsymbol{S}$ is defined as

$$E(NPV(\boldsymbol{S})) = \sum_{\boldsymbol{\omega}' \in \Omega} (NPV(\boldsymbol{S}(\boldsymbol{\omega}')) \times p(\boldsymbol{\omega} = \boldsymbol{\omega}')), \quad (11)$$

where Ω is the domain of $\boldsymbol{\omega}$ and $p(\boldsymbol{\omega} = \boldsymbol{\omega}')$ is the probability of obtaining $\boldsymbol{\omega}'$ from $\boldsymbol{\omega}$.

If $\boldsymbol{\omega}$ contains continuous random variables, the expected NPV of $\boldsymbol{S}$ is defined as

$$E(NPV(\boldsymbol{S})) = \int_{\Omega} NPV(\boldsymbol{S}(\boldsymbol{\omega}')) \times p(\boldsymbol{\omega}') \mathrm{d}\boldsymbol{\omega}', \quad (12)$$

where $p(\boldsymbol{\omega}')$ is the probability density function of $\boldsymbol{\omega}$.

Based on the above definitions, the objective of S-MRCPSPDCF is actually to find a baseline schedule $\boldsymbol{S}$ that maximizes $E(NPV(\boldsymbol{S}))$. However, because of the complexity of the random matrix $\boldsymbol{\omega}$ and the function $NPV(\boldsymbol{S}(\boldsymbol{\omega}'))$, in general, it is impossible to calculate the expected NPV directly through (11) or (12). Therefore, in the proposed approach, we use Monte Carlo simulation to estimate the expected NPV, $E(NPV(\boldsymbol{S}))$, of a baseline schedule.

4 Monte-Carlo Ant Colony System Algorithm

Compared with existing models with the consideration of financial aspects under uncertainty[21-25], the S-MRCPSPDCF model considered in this paper has the following difficulties. First, as the model considers multiple execution modes, not only the processing order of activities but also the selection of execution modes has to be addressed. Consequently, the computational complexity of the problem significantly increases. For a problem with n activities, if each activity is associated with m alternative execution modes, the computational complexity becomes $O(m^n \times (n-1)!)$. Second, as the S-MRCPSPDCF model takes account of resource constraints and uncertainty in durations and cash flows, given a baseline schedule, its expected NPV cannot be computed directly through certain expressions. The expected NPV is estimated by Monte Carlo simulation. However, implementing a comprehensive MC simulation is time consuming. Therefore, it is impractical to run comprehensive MC simulation for the estimation of every solution built by the algorithm.

In order to find a baseline schedule that maximizes the expected NPV, this paper proposes an ACO approach. ACO is characterized by dispatching a group of artificial ants to build solutions to the problem iteratively using pheromone and heuristic information. In this paper, we adopt the ACS algorithm[27], which is

by now one of the best-performed ACO variants. ACS is suitable for solving the considered S-MRCPSPDCF as it has a special feature that only the information of the best-so-far solution is used in pheromone updating. With this feature, the expected NPVs of other non-best-so-far solutions are not necessary to be estimated accurately. Only a rough MC simulation is already enough for distinguishing the best-so-far solution. In this way, the algorithm does not need to run comprehensive MC simulations for every solution, and thus the computational cost is reduced.

As the proposed algorithm couples MC simulation with ACS, we denote it as MC-ACS. The MC-ACS algorithm is composed of the following procedures:

Step 1: *Initialization.* All the parameters, heuristic information and pheromone values of the algorithm are initialized at the beginning of the algorithm.

Step 2: *Solution Construction.* Each ant in the algorithm sets out to construct a baseline schedule using pheromone and heuristic information. During the construction process, the local pheromone updating rule is applied after each move of the ants.

Step 3: *Evaluation and Comparison.* The expected NPVs of the baseline schedules built by the ants are evaluated and compared based on Monte Carlo simulation. If any of the schedules found in the current iteration outperforms the previous best-so-far schedule in the comparison, the best-so-far schedule found by the algorithm is updated.

Step 4: *Global Pheromone Updating.* The pheromone values corresponding to the best-so-far schedule are updated.

Step 5: *Terminal Test.* If the algorithm has met the terminal condition, the algorithm is terminated and the best-so-far schedule is returned. Otherwise, go to step 2.

The overall flowchart of the algorithm is shown in Fig.4. In general, the above procedures include three main operations: 1) Solution Construction — how to build baseline schedules; 2) Evaluation and Comparison — how to evaluate and compare the performance of the schedules built by the ants; and 3) Pheromone Management — how to manage the pheromone values in the algorithm, including the initialization of pheromone values in step 1, the local pheromone updating rule used in step 2, and the global pheromone updating rule used in step 4. The details for these three operations are shown in the following subsections.

4.1 Solution Construction

Let us denote the baseline schedule generated by the l-th artificial ant as

$$\boldsymbol{S}^{(l)} = \begin{pmatrix} (a_{p1}^{(l)}, a_{p2}^{(l)}, \ldots, a_{p_n}^{(l)}) \\ (k_{p1}^{(l)}, k_{p2}^{(l)}, \ldots, k_{p_n}^{(l)}) \end{pmatrix}. \tag{13}$$

The schedule defines an order of activities $(a_{p1}^{(l)}, a_{p2}^{(l)}, \ldots, a_{p_n}^{(l)})$ and the selection of the execution modes $(k_{p1}^{(l)}, k_{p2}^{(l)}, \ldots, k_{pn}^{(l)})$. It can be actually mapped

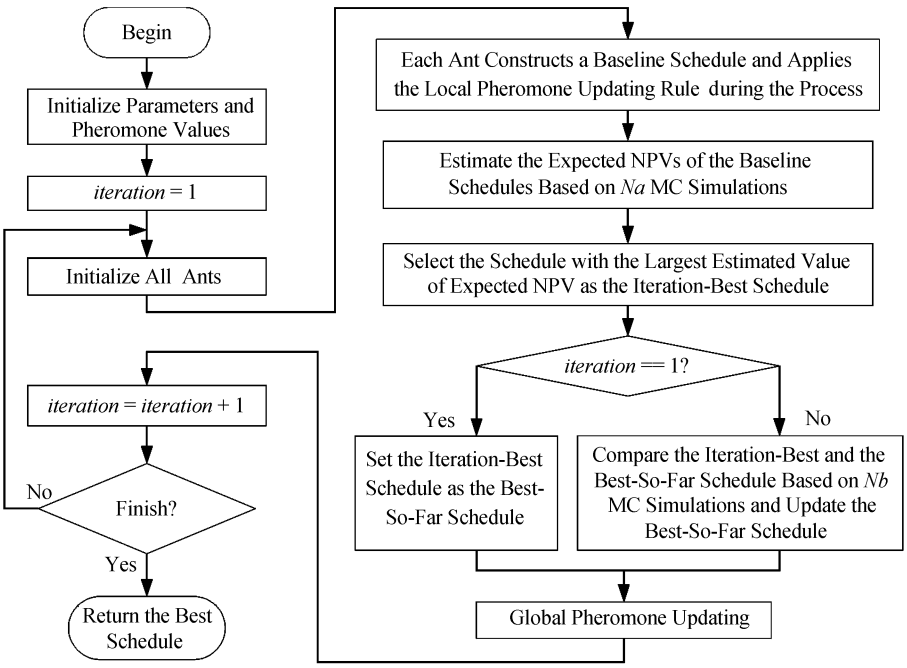

Fig.4. Flowchart of the MC-ACS algorithm.

to a list of execution modes

$$\boldsymbol{S}^{(l)} = (m_{p_1 k_{p1}^{(l)}}, m_{p_2 k_{p2}^{(l)}}, \ldots, m_{p_n k_{pn}^{(l)}}). \quad (14)$$

Therefore, to build a baseline schedule, the ants in MC-ACS add execution modes to the list step by step using pheromone and heuristic values until the schedule has been completed. To facilitate description, here we first define the pheromone and heuristic values. Then the procedures of building baseline schedules will be described in detail.

4.1.1 Definition of Pheromone and Heuristic Values

ACO algorithms are characterized by using pheromone and heuristic values to build solutions. Pheromone is a kind of memory of the previous search experiences, and heuristic is some problem-based information. In MC-ACS, pheromone values are defined on the connections between execution modes, and heuristic values are defined on execution modes. Suppose an ant has chosen a mode m_{uv} in the list, the pheromone value for choosing mode m_{ij} as the next execution mode in the list is given by $\tau(uv, ij)$. The heuristic value for choosing mode m_{ij} is given by $\eta(ij)$. The heuristic value is defined based on the expected discounted costs of execution modes. Thus we term it the EDC heuristic. Assuming $\bar{d}_{ij}$ and $\bar{c}_{ij}$ are the expected values of d_{ij} and c_{ij}, $\eta(ij)$ can be computed by

$$\eta(ij) = \frac{1}{\bar{c}_{ij} \times \exp(-\alpha \times \bar{d}_{ij})}. \quad (15)$$

With this definition, the EDC heuristic favors the modes with low expected discounted cost.

4.1.2 Eligible Set

A feasible baseline schedule has to satisfy the precedence constraints of the project. To guarantee the feasibility of the schedules, MC-ACS applies the technique of eligible sets.

In the algorithm, each ant maintains an eligible set of feasible modes that satisfy the precedence constraints. We denote the eligible set of the l-th ant as $eligible_l$. For a mode m_{ij}, if its corresponding activity a_i has not been selected in the baseline schedule and all direct predecessor activities of a_i have been selected, then m_{ij} is considered to be eligible. During the construction process, only the execution modes in the eligible set can be chosen. In this way, the schedules generated by ants can always satisfy the precedence constraints.

4.1.3 Construction Process

Let m_{00} be a dummy mode that represents the beginning of the project. At the beginning of the construction process, every ant in the algorithm starts with the dummy mode m_{00}.

Suppose that in the $(t-1)$-th step, the l-th ant has selected the execution mode m_{uv}, then in the t-th step, the selection rule for choosing the next execution mode is given by (16) and (17).

$$m_{ij} = \begin{cases} \arg\max_{m_{ij} \in eligible_l} \{\tau(uv, ij) \times (\eta(ij))^{\beta}\}, \\ \qquad\qquad \text{if } q \leqslant q_0, \\ \text{implement roulette wheel selection,} \\ \qquad\qquad \text{otherwise,} \end{cases} \quad (16)$$

$$p(m_{ij}) = \begin{cases} \dfrac{\tau(uv, ij) \times (\eta(ij))^{\beta}}{\sum\limits_{m_{xy} \in eligible_l} \tau(uv, xy) \times (\eta(xy))^{\beta}}, \\ \qquad\qquad \text{if } m_{ij} \in eligible_l, \\ 0, \qquad\qquad \text{otherwise.} \end{cases} \quad (17)$$

In (16), a random number $q \in [0, 1]$ is generated and compared with parameter q_0. If $q \leqslant q_0$, the mode m_{ij} that has the largest value of $\tau(uv, ij) \times (\eta(ij))^{\beta}$ among all the modes in the eligible set is selected. Here η is a parameter to weigh the influence of heuristic values. Otherwise, if $q > q_0$, the roulette wheel selection (RWS) scheme is adopted. In the RWS, as shown by (17), the probability of choosing m_{ij} is in direct proportion to the value of $\tau(uv, ij) \times (\eta(ij))^{\beta}$.

To summarize, the pseudo-code of how ants build baseline schedules is shown in Fig.5.

```
procedure Solution_Construction
1   Each ant selects the dummy mode m00 as the starting
    mode;
2   Initialize the eligible set of all ants;
3   for t = 1 to n
4       for l = 1 to M
5           The l-th ant selects a mode m_ij from its eligible
            set based on pheromone and heuristic values;
6           a_{p_t}^{(l)} ← a_i;
7           k_{p_t}^{(l)} ← j;
8           Update the eligible set of te l-th ant based on
            precedence relations;
9           Local pheromone updating;
10      end for
11  end for
end procedure
```

Fig.5 Pseudo-code of constructing baseline schedules.

4.2 Monte Carlo Based Evaluation and Comparison

Since there are random durations and costs in the S-MRCPSPDCF model, it is impossible to evaluate the

expected NPV of cash flows for a baseline schedule $\boldsymbol{S}$ directly. In order to evaluate and compare the performance of the schedules built by ants, the MC-ACS algorithm has to use MC simulation to estimate the expected NPVs.

Suppose the sample size in the simulation is N, to estimate the expected NPV of a baseline schedule $\boldsymbol{S}$, the algorithm first randomly samples N scenarios $\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \ldots, \boldsymbol{\omega}^{(N)}$ from $\boldsymbol{\omega}$. Then for the i-th scenario, it is possible to compute the NPV of the actual schedule $\boldsymbol{S}(\boldsymbol{\omega}^{(i)})$ according to Subsections 2.1 and 2.2. So the expected NPV of $\boldsymbol{S}$ can be estimated by

$$E(NPV(\boldsymbol{S})) \approx \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{S}(\boldsymbol{\omega}^{(i)}).NPV. \tag{18}$$

The pseudo-code of the MC simulation is given in Fig.6.

```
procedure MC_Simulation(S)
1  Randomly sample N scenarios ω^(1), ω^(2), ..., ω^(N) from
   ω;
2  E(NPV(S)) ← 0;
3  for i = 1 to N
4       SSGS (S, ω^(i));
5       Evaluate S(ω^(i)).NPV;
6       E(NPV(S)) ← E(NPV(S)) + S(ω^(i)).NPV;
7  end for
8  E(NPV(S)) ← E(NPV(S))/N;
end procedure
```

Fig.6. Pseudo-code of evaluating a baseline schedule $\boldsymbol{S}$ using N simulations.

In general, a larger sample size in the simulation will lead to a more accurate estimation of the expected NPV. However, the computational cost of simulation will also increase. Therefore, how to balance the trade-off between the estimation accuracy and computational cost is a key concern for the algorithm. Different from other ACO variants, the ACS algorithm[27] has a special characteristic that only the information of the best-so-far solution is used to update pheromone values. Therefore, for the rest worse solutions, it is not necessary to estimate their expected NPVs accurately.

Based on this feature of ACS, inspired by the simulation ACO algorithm proposed by Gutjahr for PTSP[31], this paper applies a two-phase simulation method to evaluate and compare the baseline schedules built by ants in the MC-ACS algorithm. In the first phase, a small number *Na* of random scenarios are realized from $\boldsymbol{\omega}$. The expected NPVs of all the baseline schedules built in this iteration are roughly estimated based on the simulations on these *Na* scenarios. The schedule that has the maximal estimated expected NPV is chosen as the iteration-best schedule. In the second phase, a relative larger number *Nb* of random scenarios are realized. The iteration-best schedule and the current best-so-far schedule then undergo a more accurate comparison based on the simulations on these *Nb* scenarios. The winner of the comparison becomes the best-so-far schedule found by the algorithm. In this way, the NPV estimations of most schedules built by the algorithm are only based on rough MC simulations with a small number *Na* of scenarios. Thus the computational cost is reduced.

4.3 Pheromone Updating

At the beginning of the algorithm, all pheromone values on the connections between modes are initialized to $\tau_0 = 1$. During the course of the MC-ACS algorithm, pheromone values are updated by two rules: the local pheromone updating rule and the global updating rule.

The local updating rule is applied immediately after the selection of modes (line 9 of the pseudo-code in Fig.5). If an ant selects the mode m_{uv} in the $(t-1)$-th step and selects mode m_{ij} in the t-th step, the pheromone value on the connection between m_{uv} and m_{ij} is updated by

$$\tau(uv, ij) \leftarrow (1-\xi)\tau(uv, ij) + \xi\tau_0, \tag{19}$$

where $\xi \in [0, 1]$ is a parameter. The function of local pheromone updating is actually to reduce the pheromone values on the selected connections so that the following ants will have larger probabilities to explore other unselected connections.

After distinguishing the best-so-far schedule at the end of each iteration, the global updating rule is applied to reinforce the connections used by the best-so-far schedule. If m_{uv} is followed by m_{ij} in the best-so-far schedule, the pheromone value on the connection between m_{uv} and m_{ij} is updated by

$$\tau(uv, ij) \leftarrow (1-\xi)\tau(uv, ij) + \xi\Delta^{bs}, \tag{20}$$

where Δ^{bs} is defined based on the estimated expected NPV of the best-so-far schedule $\boldsymbol{S}^{bs}$ as follows

$$\Delta^{bs} = \Big(\frac{\tilde{E}(NPV(\boldsymbol{S}^{bs})) - \mathit{firstWorst}}{\mathit{firstBest} - \mathit{firstWorst}} + 1\Big) \times \psi. \tag{21}$$

Here the estimated expected NPV $\widetilde{E}(NPV(\boldsymbol{S}^{bs}))$ is calculated in the second-phase simulation based on *Nb* random scenarios, *firstBest* is the estimated expected NPV of the iteration-best schedule in the first iteration, *firstWorst* is the smallest estimated expected NPV obtained in the first iteration, and ψ ($\psi \geqslant 1$) is a parameter to control the scale of pheromone amount. By implementing the global pheromone updating rule, addition pheromones are added on the best-so-far schedule $\boldsymbol{S}^{bs}$.

As a result, the components on $\boldsymbol{S}^{bs}$ will be more attractive for the following ants, making the search process gradually converge to $\boldsymbol{S}^{bs}$.

5 Experimental Results and Comparison Studies

5.1 Test Instances

To validate the proposed algorithm, 33 instances are randomly generated. The instances are of 11 different sizes (numbers of activities), i.e., $n \in \{13, 16, 18, 28, 36, 40, 48, 58, 60, 80, 98\}$. Each size corresponds to three instances. The three instances belong to three groups: group a, group b, and group c. The degree of uncertainty is different in different groups. For the sake of convenience, we name the three instances with 13 activities as Ins13_a, Ins13_b, and Ins13_c, respectively, and the other instances are named in the same way.

In these instances, the project networks are derived from our previous work[10]. These networks are generated based on the random activity network generator[44]. Each activity of the project is randomly associated with one to five alternative execution modes. For each execution mode m_{ij} ($i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, |M_i|$), the number of resources consumed per time period is randomly given. The duration d_{ij} of m_{ij} is assumed to be integer valued random variable of uniform distribution within an interval $[(1-\lambda)\bar{d}_{ij}, (1+\lambda)\bar{d}_{ij}]$, where $\bar{d}_{ij}$ is the expectation of d_{ij}. The cost c_{ij} of m_{ij} is assumed to be real valued random variable of normal distribution $N(\bar{c}_{ij}, \sigma^2)$, where $\bar{c}_{ij}$ is the expectation of c_{ij}. The expectations of durations $\bar{d}_{ij}$ and costs $\bar{c}_{ij}$ are all randomly given. For the instances belonging to group a, group b, and group c, we set ($\lambda = 5\%$, $\sigma = \bar{c}_{ij}$), ($\lambda = 8\%$, $\sigma = 5\bar{c}_{ij}$), and ($\lambda = 10\%$, $\sigma = 10\bar{c}_{ij}$), respectively. In other words, the instances in group b have a higher degree of uncertainty than the ones in group a, and the instances in group c have the highest degree of uncertainty.

5.2 Parameter Configuration

The MC-ACS algorithm has the following parameters: the number of ants M, the evaporation rate ξ in (19) and (20), the parameter β to weigh the influence of heuristics values in (16) and (17), the probability q_0 of directly selecting the execution mode with the largest heuristic and pheromone values in (16), the parameter ψ to control the scale of pheromone values in (21), and the sample sizes Na and Nb in MC simulations.

The first five parameters are set as follows: $M = 10$, $\xi = 0.1$, $\beta = 1$, $q_0 = 0.9$, and $\psi = 20$. These configurations are based on our previous studies of ACO for solving the deterministic project scheduling problem without uncertainties[10]. We find that these configurations still contribute to good performance with respect to S-MRCPSPDCF.

Here we focus on analyzing the performance of the algorithm under different sample sizes Na and Nb. The results of six configurations for Na and Nb on 33 instances are reported in Table 1. In the table, Nb (I) means a special scheme to increase Nb linearly from 1 to 10 with the growing iteration number[32]. In the experiment, the algorithm with each configuration is run for 30 times. Each run is terminated when the total number of sampled scenarios has reached a predefined maximum value MAX. In addition, to compare the performance of different runs, 10 000 scenarios are randomly sampled before executing the algorithm. For every schedule outputted by the algorithm, its NPV expectation is estimated based on the simulation on these 10 000 scenarios. The table reports the average results of each configuration on 30 runs.

From the table, it can be seen that with $Na = Nb = 1$, the performance of the algorithm is very unstable. This is because only a single scenario is far from enough for estimating the real quality of baseline schedules. In this case, the algorithm has insufficient information to distinguish the best-so-far solution. Thus it may regard some sub-optimal solutions which are only good at individual scenarios as the best-so-far solution. On the other hand, if $Na \geqslant 2$ or $Nb \geqslant 10$, although the simulation is more accurate, the computational cost in each simulation is also increased. Therefore, under the same evaluation number MAX, a larger Nb will lead to less iterations and thus impair the performance of the algorithm. Overall, the configuration with $Na = 1$ and increasing Nb is able to achieve the best performance.

To understand why the above configuration is effective, we further analyze the successful rate of the MC simulations under different sample sizes. At the early stages of evolutions (from the iterations 1∼20), we randomly choose 100 pairs of baseline schedules built by the ants. We compare these schedules using rough MC simulations with sample sizes 1, 2, 5, and 10, respectively. If the comparison result of the rough MC simulation for a pair of the baseline schedules is the same as the result yielded by the MC simulation with 10 000 sample size, we consider this rough simulation is successful. Otherwise, the simulation is considered failed. We also study the successful rates of the MC simulations at the later stages of evolutions (iterations 201∼220) or on 100 pairs of the iteration-best and the best-so-far solutions. The results are shown in Table 2. According to the table, at the early stage, because the differences among the performance of the baseline

Table 1. Performance of MC-ACS under Different Sample Sizes

Instance	*MAX*	$Na=1$ $Nb=1$	$Na=1$ $Nb=5$	$Na=1$ $Nb=10$	$Na=1$ $Nb=20$	$Na=2$ $Nb=5$	$Na=1$ Nb (I)
Ins13_*a*	10 000	2 333	2 344	2 312	2 292	2 188	**2 359**
Ins13_*b*	10 000	2 430	2 530	2 543	2 520	2 507	**2 569**
Ins13_*c*	10 000	2 315	**2 513**	2 482	2 404	2 442	2 466
Ins16_*a*	10 000	2 957	**2 987**	2 966	2 920	2 931	2 972
Ins16_*b*	10 000	1 102	1 172	1 130	1 009	1 110	**1 199**
Ins16_*c*	10 000	3 642	3 725	3 694	3 673	3 699	**3 730**
Ins18_*a*	10 000	2 485	2 511	**2 514**	2 491	2 398	2 508
Ins18_*b*	10 000	3 827	3 862	3 756	3 637	3 589	**3 947**
Ins18_*c*	10 000	3 273	3 434	3 401	3 361	3 288	**3 447**
Ins28_*a*	10 000	**3 388**	3 193	2 957	2 825	2 826	3 386
Ins28_*b*	10 000	579	741	515	355	223	**818**
Ins28_*c*	10 000	5 668	5 933	6 036	5 982	5 798	**6 098**
Ins36_*a*	20 000	**3 097**	2 790	2 299	2 213	2 330	2 978
Ins36_*b*	20 000	2 717	2 699	2 473	2 379	2 401	**3 017**
Ins36_*c*	20 000	1 381	1 834	1 479	980	881	**2 206**
Ins40_*a*	20 000	8 420	**9 347**	9 149	8 891	8 719	9 279
Ins40_*b*	20 000	−1 242	−1 569	−1 718	−2 274	−2 410	**− 754**
Ins40_*c*	20 000	−2 921	21	−28	−507	−1 030	**451**
Ins48_*a*	20 000	**1**	−958	−1 452	−2 859	−2 691	−582
Ins48_*b*	20 000	**1 828**	853	860	−27	−130	1 417
Ins48_*c*	20 000	5 642	6 799	6 499	6 148	5 710	**7 147**
Ins58_*a*	30 000	7 408	9 304	9 405	9 286	9 001	**9 545**
Ins58_*b*	30 000	4 844	8 198	7 919	6 947	6 130	**8 544**
Ins58_*c*	30 000	7 236	9 223	**9 382**	9 023	9 210	9 289
Ins60_*a*	30 000	−199	−356	−658	−1 947	−2 134	**448**
Ins60_*b*	30 000	2 800	5 305	498	3 899	3 139	**5 467**
Ins60_*c*	30 000	−291	5 276	4 833	3 888	3 801	**5 499**
Ins80_*a*	40 000	−3 147	−1 215	−2 321	−3 086	−2 810	**−73**
Ins80_*b*	40 000	3 499	5 342	4 713	3 038	2 751	**6 101**
Ins80_*c*	40 000	−3 800	**4 231**	2 140	666	49	3 810
Ins98_*a*	50 000	67	2 562	2 546	360	−239	**3 469**
Ins98_*b*	50 000	−6 643	−2 065	−3 409	−4 012	−5 012	**−1 116**
Ins98_*c*	50 000	−7 122	314	−515	−1 759	−2 310	**1 594**

Note: The best results yielded by the most suitable sample size are marked in bold.

Table 2. Analysis of the Successful Rates of Rough Simulations

Stages	Comparison Objects	Sample Size (%)			
		1	2	5	10
Iterations 1∼20	Ordinary solutions	92	96	99	100
Iterations 1∼20	Iteration-best and best-so-far solutions	84	89	94	98
Iterations 201∼220	Ordinary solutions	86	89	95	99
Iterations 201∼220	Iteration-best and best-so-far solutions	56	72	84	93

schedules built by the ants are significant, even the MC simulation with only one sample already contributes to a 92% successful rate. At the late stage, the MC simulation with only one sample can still contribute to an 86% successful rate. Therefore, setting $Na = 1$ is already enough to correctly tell apart the better baseline schedule in the comparison in most cases. On the other hand, although the MC simulation with one sample achieves an 84% successful rate at the early stage for the comparisons between the iteration-best and the best-so-far solutions, its successful rate drops to 56% at the late stage. As the difference between the performance of the iteration-best and the best-so-far solutions is usually slight at the late stage as the algorithm converges, we need to use the MC simulation with 10 samples to guarantee a 93% successful rate. These results explain why increasing Nb from 1 to 10 linearly can result in good performance.

5.3 Parameter Configuration

By now, to the best of our knowledge, the stochastic form of the MRCPSPDCF model has not been considered in the literature. In order to demonstrate the effectiveness of the proposed algorithm, we implement three other approaches to solve the S-MRCPSPDCF in the comparison. The basic idea of these three approaches are derived from some existing work on project scheduling[10] and optimization under uncertainty [31,39-40].

First, according to Bianchi *et al.*[39], when dealing with stochastic optimization problems, a possible approach is to design a computable ad hoc function to approximate the actual objective function. In traditional deterministic project scheduling models, the environmental parameters such as durations and costs are given deterministically. As the execution of a project is usually subject to uncertainty, the deterministic models can be actually regarded as using some predetermined setting of environmental parameters (e.g., the expected value of durations and costs) to approximate the actual objective function of the scheduling problem. Therefore, in the first approach, we apply the ACO algorithm for deterministic MRCPSPDCF[10] and use the expected durations and costs as the predetermined environmental parameters to solve S-MRCPSPDCF approximately. Since the algorithm is based on deterministic models, we term it D-ACO in this subsection.

Second, according to Jin and Branke[40], to deal with stochastic optimization problems, there is an implicit averaging method in which the population size of an algorithm is increased to reduce the influence of noise. Following this idea, we modify the proposed MC-ACS algorithm to use a large population size $M = 100$ and use the MC simulation with a single random scenario (i.e., $Na = Nb = 1$) to estimate the performance of baseline schedules. Since the algorithm has large population size, we denote it as MC-ACS-LPS for short.

Finally, in [31], a simulation ACO (S-ACO) algorithm has been developed for solving a traveling salesman problem with time windows (TSPTW) in the case of stochastic service times. In the third approach, we follow [31] to implement an S-ACO algorithm for S-MRCPSPDCF. In order to make S-ACO suitable for the considered problem, we use the same pheromone and heuristic definitions as the proposed MC-ACS in S-ACO. The main difference between S-ACO and MC-ACS lies in the implementation of ACO. While S-ACO follows the conventional ACO procedure that uses the roulette wheel selection rule in solution construction and uses the pheromone evaporation rule in pheromone management, MC-ACS follows the procedure of the ACS algorithm[27]. Different from other ACO variants, ACS only uses the information of the best-so-far solution to update pheromone values. Therefore, we only need to tell apart the best-so-far solution and the rankings of all the other solutions can be ignored. As a result, in the evaluation and comparison procedure, we can perform rough estimations to reduce the computational cost.

In the experiment, the parameter configuration of MC-ACS has been given in Subsection 5.2, and the parameter configurations of the three approaches for comparison are given in Table 3. For each instance, each algorithm is run for 30 times. Each run is terminated when the total number of sampled scenarios has reached a predefined maximum value MAX, which has been given in Table 1. Similar to the experiments in Subsection 5.2, for each instance, 10 000 scenarios were first randomly sampled. The performance of the schedules outputted by different algorithms is compared based on these 10 000 random scenarios.

Table 3. Parameter Settings for the Algorithms in Comparison

Algorithm	**Parameter Configuration**
D-ACO	$M = 10$, $\xi = 0.1$, $\beta = 1$, $q_0 = 0.9$, and $\psi = 20$ The parameters are set according to [9].
MC-ACS-LPS	$M = 100$, $\xi = 0.1$, $\beta = 1$, $q_0 = 0.9$, $\psi = 20$, $Na = Nb = 1$. The parameters are set according to the idea[39] of using larger population size instead of running multiple random sampling.
S-ACO	$M = 50$, $\rho = 0.1$, the sampling number in MC simulation is increased from 1 to 10 linearly with the growing number of generations. The parameters are set according to [30].

In Table 4, the results of MC-ACS, D-ACO, MC-ACS-LPS and S-ACO averaged over 30 runs on the 33 S-MRCPSPDCF instances are tabulated. According to the table, it can be seen that the proposed MC-ACS algorithm outperforms the other approaches in the comparison in most instances. Out of the 33 instances, MC-ACS obtains better average expected NPVs than D-ACO, MC-ACS-LPS and S-ACO on 31, 28 and 30 instances, respectively, and achieves the best results on 26 instances. In addition, we also run two-sample t-tests to analyze if the differences between the results are significant. The comparison results are listed in Table 5. According to the t values in Table 5, MC-ACS yields significantly better results than D-ACO, MC-ACS-LPS and S-ACO on 26, 21 and 23 instances, respectively. In particular, for the instances with 60 activities or more, MC-ACS obtains significantly better results in all cases.

Table 4. Experimental Results of MC-ACS, D-ACO, MC-ACS-LPS and S-ACO

Algorithm	Size	Group a	Group b	Group c
MC-ACS	13	2 359	2 569	**2 466**
D-ACO		2 215	**2 577**	2 420
MC-ACS-LPS		2 344	2 392	2 263
S-ACO		**2 420**	2 563	2 432
MC-ACS	16	**2 972**	1 199	**3 730**
D-ACO		2 942	1 137	3 670
MC-ACS-LPS		2 941	1 172	3 617
S-ACO		2 935	**1 208**	3 690
MC-ACS	18	2 508	**3 947**	**3 447**
D-ACO		**2 518**	3 353	3 360
MC-ACS-LPS		2 512	3 870	3 264
S-ACO		2 415	3 834	3 318
MC-ACS	28	3 386	**818**	**6 098**
D-ACO		2 663	488	5 420
MC-ACS-LPS		**3 477**	407	5 597
S-ACO		3 063	768	5 746
MC-ACS	36	2 978	**3 017**	**2 206**
D-ACO		2 415	1 715	562
MC-ACS-LPS		**3 099**	2 689	1 293
S-ACO		2 568	2 211	1 846
MC-ACS	40	**9 279**	**−754**	**451**
D-ACO		7 601	−2 582	−1 661
MC-ACS-LPS		8 414	−1 585	−1 697
S-ACO		8 562	−968	−314
MC-ACS	48	−582	1 417	**7 147**
D-ACO		−1 228	956	5 529
MC-ACS-LPS		−125	**1 638**	4 951
S-ACO		**−97**	25	6 486
MC-ACS	58	**9 545**	**8 544**	**9 289**
D-ACO		8 547	5 059	7 129
MC-ACS-LPS		6 913	5 180	7 357
S-ACO		9 453	7 689	8 592
MC-ACS	60	**448**	**5 467**	**5 499**
D-ACO		−1 479	3 842	2 865
MC-ACS-LPS		−939	3 671	755
S-ACO		−442	4 103	4 501
MC-ACS	80	**−73**	**6 101**	**3 810**
D-ACO		−3 397	1 602	−628
MC-ACS-LPS		−2 872	2 743	−2 733
S-ACO		−2 433	3 738	119
MC-ACS	98	**3 469**	**−1 116**	**1 594**
D-ACO		1 278	−5 584	−1 830
MC-ACS-LPS		−1 589	−7 202	−3 972
S-ACO		263	−3 197	−1 302

Note: The results are averaged on 30 runs. The best results yielded by the algorithms are marked in bold.

Table 5. t-test Comparison between MC-ACS and Other Approaches

Algorithm	Size	Group a	Group b	Group c
MC-ACS−D-ACO	13	3.433#	−0.505	1.069
MC-ACS−MC-ACS-LPS		0.580	3.640#	3.429#
MC-ACS−S-ACO		−2.656*	0.290	0.834
MC-ACS−D-ACO	16	0.700	0.772	1.161
MC-ACS−MC-ACS-LPS		0.747	0.362	4.581#
MC-ACS−S-ACO		1.210	−0.159	2.252#
MC-ACS−D-ACO	18	−0.670	5.034#	3.026#
MC-ACS−MC-ACS-LPS		−0.271	0.767	4.688#
MC-ACS−S-ACO		3.672#	1.294	4.474#
MC-ACS−D-ACO	28	4.622#	2.011#	6.306#
MC-ACS-MC−ACS-LPS		−0.757	2.756#	4.331#
MC-ACS−S-ACO		2.737#	0.362	5.312#
MC-ACS−D-ACO	36	2.852#	5.018#	5.880#
MC-ACS−MC-ACS-LPS		−0.537	1.806	2.599#
MC-ACS−S-ACO		2.231#	6.053#	1.738
MC-ACS−D-ACO	40	6.121#	4.002#	6.018#
MC-ACS−MC-ACS-LPS		3.516#	2.001#	5.705#
MC-ACS−S-ACO		4.929#	0.635	3.521#
MC-ACS−D-ACO	48	1.540	1.11	5.242#
MC-ACS−MC-ACS-LPS		−1.000	−0.546	6.261#
MC-ACS−S-ACO		−2.048*	3.691#	3.756#
MC-ACS−D-ACO	58	4.655#	8.613#	6.502#
MC-ACS−MC-ACS-LPS		4.142#	8.003#	6.783#
MC-ACS−S-ACO		0.473	3.819#	2.460#
MC-ACS−D-ACO	60	5.154#	5.446#	6.376#
MC-ACS−MC-ACS-LPS		2.611#	4.243#	6.560#
MC-ACS−S-ACO		2.604#	4.771#	2.931#
MC-ACS−D-ACO	80	7.381#	8.818#	7.761#
MC-ACS−MC-ACS-LPS		4.339#	5.166#	11.090#
MC-ACS−S-ACO		5.474#	5.940#	7.244#
MC-ACS−D-ACO	98	3.507#	5.728#	5.996#
MC-ACS−MC-ACS-LPS		5.818#	10.040#	11.760#
MC-ACS−S-ACO		6.094#	4.387#	5.604#

Note: "#" means the result of MC-ACS is significantly better than the result obtained by the algorithm used in comparison based on a two-sample t-test with 58 degree of freedom at the 0.05 level of significance. "$*$" means the result of MC-ACS is significantly worse than the result obtained by the algorithm used in comparison based on a two-sample t-test with 58 degree of freedom at the 0.05 evel of significance.

These results reveal that the proposed algorithm is effective.

In stochastic optimization problems, we usually concern not only the average results, but also the frequency distribution of the results. As the schedules outputted by the algorithms in comparison have been simulated on 10 000 randomly sampled scenarios, we here further study the distributions of NPVs on these 10 000 scenarios. For each algorithm, the best baseline schedule found by the algorithm over the 30 runs is considered. The histograms of the NPVs of the best schedules found by MC-ACS, D-ACO, MC-ACS-LPS and S-ACO on some representative instances are plotted in Fig.7. According to the histograms, the proposed MC-ACS algorithm can find schedules with larger NPVs steadily. This phenomenon is even more obvious in large-size

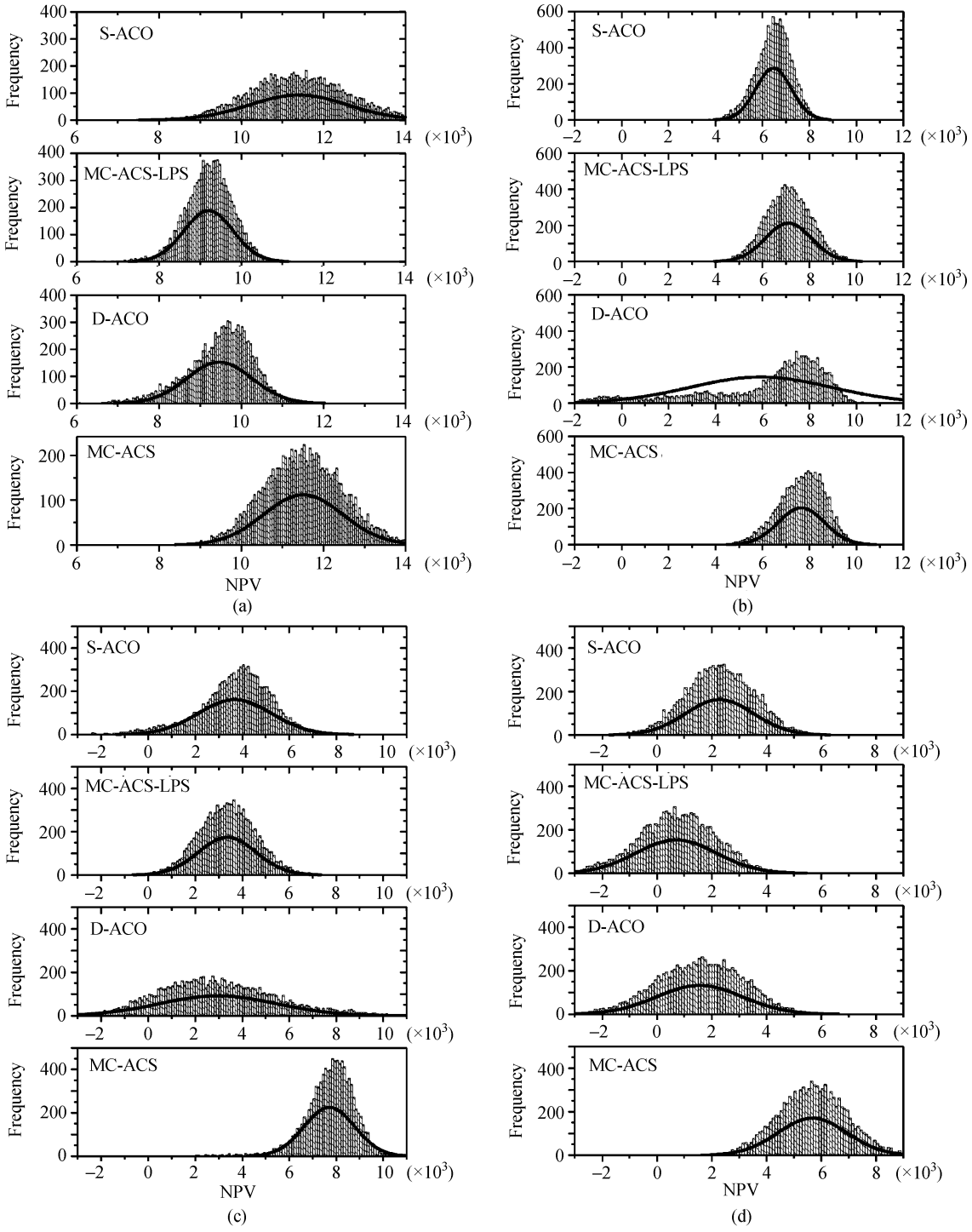

Fig.7. Histograms of the NPVs of the best schedules found by S-ACO, MC-ACS-LPS, D-ACO and MC-ACS based on 10 000 randomly sampled scenarios. (a) Ins58_*c*. (b) Ins60_*c*. (c) Ins80_*c*. (d) Ins98_*c*.

instances such as Ins80_*c* (Fig.7(c)) and Ins98_*c* (Fig.7(d)). In Ins80_*c*, while the NPVs of the best schedule found by MC-ACS are around 5 000 to 10 000 with the highest frequency on around 8 000, the NPVs of the schedules found by other algorithms are only around −2 000 to 7 000. In Ins98_*c*, the NPVs of the schedule found by MC-ACS are around 3 000 to 8 000, but the ones found by other algorithms are usually smaller than 5 000. These results also reveal that the proposed algorithm is effective.

6 Conclusions

An MC-ACS algorithm has been proposed to solve S-MRCPSPDCF. As the considered scheduling model contains random variables, MC-ACS uses the MC simulation to estimate the expected NPV of cash flows. In the experiment, it is found that only a rough simulation based on a small number of random scenarios is already enough for balancing the simulation accuracy and computational cost of the algorithm. Therefore, the algorithm is effective to find promising baseline schedules. For future research, designing local search methods and developing adaptive schemes to tune the sample size will be some potential ways to further enhance the performance of the algorithm.

Acknowledgement The authors would like to thank the anonymous reviewers and the editors for their valuable comments and suggestions to improve the paper's quality.

References

[1] Brucker P, Drexl A, Möhring R, Neumann K, Pesch E. Resource-constrained project scheduling: Notation, classification, models and methods. *European Journal of Operational Research*, 1999, 112(1): 3-41.

[2] Blazewicz J, Lenstra J K, Rinnooy Kan A H G. Scheduling subject to resource constraints: Classification and complex. *Discrete Applied Mathematics*, 1983, 5(1): 11-24.

[3] Ödamar L. A genetic algorithm approach to a general category project scheduling problem. *IEEE Transactions on Systems, Man, and Cybernetics — Part C*: *Application and Reviews*, 1999, 29(1): 44-59.

[4] Tseng L Y, Chen S C. Two-phase genetic local search algorithm for the multimode resource-constrained project scheduling problem. *IEEE Transactions on Evolutionary Computation*, 2009, 13(4): 848-857.

[5] Kolisch R. Project Scheduling under Resource Constraints: Efficient Heuristics for Several Problem Classes. Physica-Verlag, 1995.

[6] Russell A H. Cash flows in networks. *Management Science*, 1970, 16(5): 357-373.

[7] Herroelen W, De Reyck B, Demeulemeester E. Project network models with discounted cash flows: A guided tour through recent developments. *European Journal of Operational Research*, 1997, 100(1): 97-121.

[8] Ulusoy G. Four payment models for the multi-mode resource constrained project scheduling problem with discounted cash flows. *Annals of Operations Research*, 2001, 102(1-4): 237-261.

[9] Mika M, Waligóra G, Węglarz J. Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. *European Journal of Operational Research*, 2005, 164(3): 639-668.

[10] Chen W N, Zhang J, Chung H S H, Huang R Z, Liu O. Optimizing discounted cash flows in project scheduling — An ant colony optimization approach. *IEEE Transactions on Systems, Man, and Cybernetics — Part C*: *Applications and Reviews*, 2010, 40(1): 64-77.

[11] Herroelen W, Leus R. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 2005, 165(2): 289-306.

[12] Li Z, Ierapetritou M. Process scheduling under uncertainty: Review and challenges. *Computers and Chemical Engineering*, 2008, 32(4-5): 715-727.

[13] Sahinidis N V. Optimization under uncertainty: State-of-the-art and opportunities. *Computers and Chemical Engineering*, 2004, 28(6-7): 971-983.

[14] Park H K. Cash flow forecasting in construction project. *KSCE Journal of Civil Engineering*, 2004, 18(3): 265-271.

[15] Yang I T, Chang C Y. Stochastic resource-constrained scheduling for repetitive construction projects with uncertain supply of resources and funding. *International Journal of Project Management*, 2005, 23(7): 546-553.

[16] Wu X, Zhou X. Stochastic scheduling to minimize expected maximum lateness. *European Journal of Operational Research*, 2008, 190(1): 103-115.

[17] Ke H, Liu B. Project scheduling problem with stochastic activity duration times. *Applied Mathematics and Computation*, 2005, 168(1): 342-353.

[18] Guo Z X, Wong W X, Leung S Y S, Fan J T, Chan S F. Genetic optimization of order scheduling with multiple uncertainties. *Expert Systems with Applications*, 2008, 35(4): 1788-1801.

[19] Long L D, Ohsato A. Fuzzy critical chain method for project scheduling under resource constraints and uncertainty. *International Journal of Project Management*, 2008, 26(6): 688-698.

[20] Daniels R L, Carrillo J E. β-robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions*, 1997, 29(11): 977-985.

[21] Creemers S, Leus R, De Reyck B, Lambrecht M. Project scheduling for maximum NPV with variable activity durations and uncertain activity outcomes. In *Proc. IEEE International Conference on Industrial Engineering and Engineering Management*, Dec. 2008, pp.183-187.

[22] Creemers S, Leus R, Lambrechts M. Scheduling Markovian PERT networks with maximum-NPV objective. Technical Report KBI_0811, Department of Decision Sciences and Information Management, KU Leuven, Belgium, 2008.

[23] Sobel M J, Szmetekovsky J G, Tilson V. Scheduling projects with stochastic activity duration to maximize expected net present value. *European Journal of Operational Research*, 2009, 198(3): 697-705.

[24] Wiesemann W, Kuhn D, Rustem B. Maximizing the net present value of a project under uncertainty. *European Journal of Operational Research*, 2010, 202(2): 356-367.

[25] Szmerekovsky J G. Single machine scheduling under market uncertainty. *European Journal of Operational Research*, 2007, 177(1): 163-175.

[26] Dorigo M, Maniezzo V, Colorni A. Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics — Part B*: *Cybernetics*, 1996, 26(1): 29-41.

[27] Dorigo M, Gambardella L M. Ant colony system: A cooperative learning approach to TSP. *IEEE Transactions on Evolutionary Computation*, 1997, 1(1): 53-66.

[28] Merkle D, Middendorf M, Schmeck H. Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*, 2002, 6(4): 333-346.

[29] Chen W N, Zhang J. An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. *IEEE Transactions on Systems, Man, and Cybernetics — Part C*:*Application and Reviews*, 2009, 39(1): 29-43.

[30] Hu X M, Zhang J, Chung H S H, Liu O, Xiao J. An intelligent testing system embedded with an ant colony optimization based test composition method. *IEEE Transactions on Sys-*

tems, Man and Cybernetics —Part C:Application and Reviews, 2009, 39(6): 659-669.

[31] Gutjahr W J. S-ACO: An ant based approach to combinatorial optimization under uncertainty. In *Proc. the 4th ANTS*, Sept. 2004, pp.238-249.

[32] Gutjahr W J. A converging ACO algorithm for stochastic combinatorial optimization. In *Proc. the 2nd Symposium on Stochastic Algorithms, Foundations and Applications*, Sept. 2003, pp.10-25.

[33] Balaprakash P. Ant colony optimization under uncertainty. Technical Report No. TR/IRIDIA/2005-028, IRIDIA, Universite Libre de Bruxelles, 2005.

[34] Brucker P, Knust S, Schoo A, Thiele O. A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 1998, 107(2): 272-288.

[35] Lee J K , Kim Y D. Search heuristics for resource constrained project scheduling. *Journal of the Operational Research Society*, 1996, 47(5): 678-689.

[36] Chiang C W, Huang Y Q, Wang W Y. Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, 2008, 19(4-5): 345-358.

[37] Józefowska J, Mika M, Różycki R, Waligóra G, Węglarz J. Simulated annealing for multi-mode resource-constrained project scheduling. *Annals of Operations Research*, 2001, 102(1-4): 137-155.

[38] Bianchi L, Dorigo M, Gambardella L M, Gutjahr W J. Metaheuristics in stochastic combinatorial optimization: A survey. Technical Report IDSIA-08-06, IDSIA, Dalle Molle Institute for Artificial Intelligence, 2006.

[39] Bianchi L, Dorigo M, Gambardella L M, Gutjahr W J. A survey on metaheuristics for combinatorial optimization. *Nat. Comput.*, 2009, 8(2): 239-287.

[40] Jin Y, Branke J. Evolutionary optimization in uncertain environments — A survey. *IEEE Transactions on Evolutionary Computation*, 2005, 9(3): 303-317.

[41] Vieira G E, Herrmann J W, Lin E. Rescheduling manufacturing systems: A framework of strategies, policies and methods. *Journal of Scheduling*, 2003, 6(1): 39-62.

[42] Shtub A, Bard J F, Globerson S. Project Scheduling: Processes, Methodologies and Economics (2nd edition). Prentice Hall, 2002.

[43] Kolisch R, Hartmann S. Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In *Handbook on Recent Advances in Project Scheduling*. Weglarz J (ed), Dordrecht, The Netherlands: Kluwer Academic Publishers, 1999, pp.197-212.

[44] Demeulemeester E, Dodin B, Herroelen W. A random activity network generator. *Operations Research*, 1993, 41(5): 972-980.

Wei-Neng Chen received the Bachelor's degree in network engineering and the Ph.D. degree in computer application technology from the Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively. He is currently a lecturer with the Department of Computer Science, Sun Yat-sen University, China. His current research interests include evolutionary computation and its applications on financial optimization, operations research and software engineering.

Jun Zhang received the Ph.D. degree from Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China, in 2002. He is currently a professor with the Department of Computer Science, Sun Yat-sen University, Guangzhou, China. He has authored 7 research books and book chapters, and over 100 technical papers in his research areas. His research interests include computational intelligence, operations research, data mining, wireless sensor networks, and power electronic circuits. Dr. Zhang received the National Science Funds for Distinguished Young Scholars from the National Natural Science Foundation of China in 2011. He received the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. Dr. Zhang is currently an Associate Editor of the IEEE Transactions on Industrial Electronics, and Associate Editor for IEEE Computational Intelligence Magazine. He is the founding and current Chair of the IEEE Guangzhou Subsection and the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapter.