Optimizing Discounted Cash Flows in Project Scheduling—An Ant Colony Optimization Approach

Wei-Neng Chen, *Student Member, IEEE*, Jun Zhang, *Senior Member, IEEE*, Henry Shu-Hung Chung, *Senior Member, IEEE*, Rui-Zhang Huang, and Ou Liu

***Abstract*—The multimode resource-constrained project-scheduling problem with discounted cash flows (MRCPSPDCF) is important and challenging for project management. As the problem is strongly nondeterministic polynomial-time hard, only a few algorithms exist and the performance is still not satisfying. To design an effective algorithm for the MRCPSPDCF, this paper proposes an ant colony optimization (ACO) approach. ACO is promising for the MRCPSPDCF due to the following three reasons. First, MRCPSPDCF can be formulated as a graph-based search problem, which ACO has been found to be good at solving. Second, the mechanism of ACO enables the use of domain-based heuristics to accelerate the search. Furthermore, ACO has found good results for the classical single-mode scheduling problems. But the utility of ACO for the much more difficult MRCPSPDCF is still unexplored. In this paper, we first convert the precedence network of the MRCPSPDCF into a mode-on-node (MoN) graph, which becomes the construction graph for ACO. Eight domain-based heuristics are designed to consider the factors of time, cost, resources, and precedence relations. Among these heuristics, the hybrid heuristic that combines different factors together performs well. The proposed algorithm is compared with two different genetic algorithms (GAs), a simulated annealing (SA) algorithm, and a tabu search (TS) algorithm on 55 random instances with at least 13 and up to 98 activities. Experimental results show that the proposed ACO algorithm outperforms the GA, SA, and TS approaches on most cases.**

***Index Terms*—Ant colony optimization (ACO), discounted cash flows, net present value, resource-constrained project-scheduling problem (RCPSP).**

I. Introduction

CASH flow means the amount of cash being received and spent during a defined period of time. Without positive cash flows, basic obligations such as payments to suppliers and payrolls cannot be met [1], [2]. In project level, even a high-profit project may turn out to be a failure if cash shortfall suddenly occurs. As such, cash flow management in project level has attracted a considerable amount of research effort in recent years [3]. A promising progress is to integrate cash flow management with the resource-constrained project-scheduling problem (RCPSP).

The classical RCPSP is a problem of finding an optimal schedule that satisfies the resource and precedence constraints and minimizes the makespan (duration) of a project. Applications of the RCPSP can be found in a broad area of industrial projects, such as house building and software development. For a comprehensive survey of the RCPSP, please refer to Herroelen *et al.* [3] and Brucker *et al.* [4].

The classical RCPSP ignores the cash flow criterion of a project. Russell [5] first introduced the cash flow criterion and proposed a model named max-NPV. In this model, a series of cash inflows and outflows occurs in the course of the project. Cash inflows include the payments for the execution of activities and cash outflows include all expenditures for resources. Time value of money is taken into account by discounting the cash flows. The difference between discounted cash inflows and outflows is referred to as the net present value (NPV). The objective function is to maximize the final NPV of the project. The max-NPV model with resource constraints becomes the RCPSP with discounted cash flows (RCPSPDCF) [6], [7]. Some recent surveys on the RCPSPDCF are given by Özdamar and Ulusoy [7], Herroelen *et al.* [3], [8], Brucker *et al.* [4], and Tavares [9].

A more general and practical type of the RCPSPDCF model considers the time–cost, time–resource, and resource–resource tradeoffs [10]. Each activity can be implemented by any one out of a finite number of alternative execution modes. In the situation of time–cost tradeoff, an execution mode with longer duration may cost less money, whilst a mode with shorter duration may cost more money. In the situation of time–resource tradeoff, an execution mode with longer duration consumes fewer resources, *vice versa*. In the situation of resource–resource tradeoff, the usage of a resource in an execution mode can substitute for the other resources. The presence of such tradeoffs leads to the multimode RCPSPDCF (MRCPSPDCF) [11].

The classical RCPSP is nondeterministic polynomial-time hard (NP-hard) as it contains the job shop problem (JSP) as a special case [12]. The computational effort increases in the RCPSPDCF by having to evaluate the NPV in a nonlinear model. Furthermore, the computational complexity of the MRCPSPDCF greatly increases as both the schedules for activities and for different modes have to be considered. Kolisch [13] has proven that the problem of finding a feasible solution for the MRCPSPDCF with more than one nonrenewable resource is already NP-complete. As the cash flow criterion is important

Manuscript received March 2, 2008; revised July 6, 2008, December 17, 2008, and March 30, 2009. This work was supported in part by the National Science Foundation of China (NSFC) under Project 60573066, the NSFC Joint Fund with Guangdong under Key Project U0835002, the National High Technology Research and Development Program ("863" Program) of China under Project 2009AA01Z208, and a departmental general research fund of the Hong Kong Polytechnic University (G-U442). This paper was recommended by Associate Editor M.-H. Lim.

W.-N. Chen and J. Zhang are with the Department of Computer Science, Sun Yat-Sen University, Guangzhou 510275, China.

H. S. H. Chung is with the Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong.

R.-Z. Huang and O. Liu are with the Hong Kong Polytechnic University, Kowloon, Hong Kong.

Digital Object Identifier 10.1109/TSMCC.2009.2027335

and the problem is difficult to solve, MRCPSPDCF has been considered as a challenging model for project management in recent years [4], [14].

For the single-mode RCPSP and RCPSPDCF, many exact and stochastic approaches have been proposed, such as zero–one integer programming [6], [15], branch-and-bound algorithm [16], [17], genetic algorithm (GA) [18], [19], ant colony optimization (ACO) [20], tabu search (TS) [21], simulated annealing (SA) [22], and specialized heuristic methods [23].

However, for the more difficult MRCPSPDCF, only a few algorithms have been developed. Ulusoy [11] considered four payment models and proposed a GA approach to the general form of MRCPSPDCF. The algorithm was tested on a set of instances with at most 51 activities and achieved good performance. However, as the algorithm does not make use of any domain-based information to accelerate the search, the search speed of the GA approach may become slow especially for large-size instances. Mika *et al.* [14] proposed an SA and a TS algorithm for the MRCPSPDCF with positive cash inflows. As the SA and TS are local search metaheuristics [14], they may sometimes get trapped in local optima. In addition, several specialized algorithms were also designed for some specific cases of the MRCPSPDCF. Icmeli and Erenguc [10] introduced a model in which crashing costs are incurred if the duration of an activity is shorter than normal. A heuristic procedure with embedded priority rules was proposed for the model. Özdamar [26] considered a housing project with various expenditure rates and a heuristic method was developed to construct and reconstruct the schedule. Ulusoy and Cebelli [27] set up a model to consider the goals from the view of both the contractor and the client and developed a double-loop GA approach.

This paper aims to take advantage of the ACO metaheuristic to propose a more effective algorithm for the intractable MRCPSPDCF. ACO is a population-based optimization algorithm proposed by Dorigo and coworkers [28]–[30] in the early 1990s in the light of how ants manage to discover the shortest path from their nest to the food source. The idea of applying ACO to the MRCPSPDCF is motivated by the following three reasons. First, the MRCPSPDCF can be naturally formulated as a graph-based search problem, which ACO has been shown to be particularly successful in solving [31]–[35]. Second, domain-based heuristics have been found to be useful for solving scheduling problems [4], [7], [12], [18]. Various effective heuristics have been proposed for the classical RCPSP [18]. The mechanism of ACO supports the use of such heuristics to improve performance. Third, the ACO algorithms for the single-mode RCPSP [36] and RCPSPDCF [20] have achieved good performance. However, to the best of our knowledge, the utility of ACO for the more difficult MRCPSPDCF is still unexplored.

In this paper, we first formulate the model of MRCPSPDCF. To make the model more realistic, indirect costs and a bonus–penalty (B-P) mechanism are taken into account. An ant colony system (ACS) algorithm is developed for the general form of the MRCPSPDCF. The algorithm is named ACS–MRCPSPDCF. ACS [32], [35] is one of the best ACO algorithms so far. To propose the ACS approach, the precedence network of the problem, which is an activity-on-arc (AoA) network [11], is converted into a mode-on-node (MoN) graph. The MoN graph becomes the construction graph for algorithm. By applying the serial schedule generation scheme (SSGS) [37], artificial ants in the algorithm build their solutions step by step on the MoN graph using pheromones and heuristic information. In order to enhance the search ability of ants, eight heuristics are introduced, including two precedence-relation-based heuristics, two time-based heuristics, two cost-based heuristics, a resource-based heuristic, and a hybrid heuristic. As there are time–cost–resource tradeoffs in MRCPSPDCF, the hybrid heuristic is able to combine the information from all aspects and performs well. Parameters of the algorithm are analyzed for balancing the exploration and exploitation ability of the search. In the experiment, the algorithm is compared with two GA approaches [11], [18], an SA algorithm and a TS algorithm [14] on 55 instances with at least 13 and up to 98 activities. Experimental results demonstrate the effectiveness of the proposed algorithm.

The rest of the paper is organized as follows. In Section II, the MRCPSPDCF model is formulated. The ACS algorithm for MRCPSPDCF is presented in detail in Section III. Experimental results are given in Section IV. The conclusion finally comes in Section V.

II. MRCPSPDCF Model

A. Problem Definition and Notation

MRCPSPDCF is a problem of finding an optimal schedule for a project so that the precedence and resource constraints are satisfied and the final NPV of the project is maximized. The basic notation for the problem is shown in Table I. The characteristics of the MRCPSPDCF are summarized as follows.

1) *Precedence constraint*—Given a project with n activities, the precedence relations of the activities are given by an AoA network $G = (E, A)$, where the node set E corresponds to the set of events and the arc set $A = \{a_1, a_2, \ldots, a_n\}$ corresponds to the set of activities. Fig. 1 shows an example of the AoA network. We add two dummy activities a_0 and a_7 to represent the beginning and the end of the project, respectively. Based on the AoA network, the precedence constraint is defined as follows. Event e_i takes place as soon as all direct predecessor activities of e_i have finished. Only after the occurrence of e_i, the direct successor activities of e_i can start.
2) *Resource constraint*—The project uses $|RR|$ types of renewable resources and $|NR|$ types of nonrenewable resources. Renewable resources are limited period by period, and nonrenewable ones are limited for the entire project.
3) *Multimode*—Each activity a_i $(i = 1, 2, \ldots, n)$ can be executed in any mode out of a finite alternative execution mode set $M_i = \{m_{i1}, m_{i2}, \ldots, m_{i|M_i|}\}$, where m_{ij} is the jth mode for a_i and $|M_i|$ is the total number of available modes for a_i. The duration and resource consumption of each execution mode are given deterministically. The duration of mode m_{ij} is d_{ij}. The mode m_{ij} consumes rr_{ij}^k units of renewable resource k, nr_{ij}^l units of nonrenewable

TABLE I
BASIC NOTATION FOR THE MRCPSPDCF MODEL

Symbol	Definition
$A=\{a_0,a_1,\ldots,a_n,a_{n+1}\}$	the set of activities; $a_1,\ldots,a_n$ represent the real activities; a_0 and a_{n+1} are dummy activities correspond to the start and the end of the project respectively; n is the number of real activities;
$E=\{e_1,\ldots,e_{\lvert E\rvert}\}$	the set of events; $\lvert E\rvert$ represents the number of events;
$pred(a_i)$ and $succ(a_i)$ $pred(e_i)$ and $succ(e_i)$	the set of direct predecessor activities and the set of direct successor activities of activity a_i or event e_i
$G=(A,E)$	the AoA graph of the project
$PE\subseteq E$	the set of the payment (milestone) events, which is a subset of E;
$M_i=\{m_{i1},\cdots,m_{i\lvert M_i\rvert}\}$	the set of execution modes in activity a_i, i=0,…,n+1; $\lvert M_i\rvert$ represents the number of modes in a_i;
RR_i	RR_i is the number of available units for the ith type of renewable resource; there are totally $\lvert RR\rvert$ types of renewable resources
PRR_i	the cost of per unit non-renewable resource in per time period
NR_i	NR_i is the number of available units for the ith type of non-renewable resource; there are totally $\lvert NR\rvert$ types of non-renewable resources
PNR_i	the cost of per unit non-renewable resource in per time period
U	the total value of the project
$W_1,\ldots,W_n$	the net worth of each activity
d_{ij}	duration of the activity i under mode j
CE_{ij}	constant expenses of the activity i under mode j
rr_{ij}^{k}	per time period usage of renewable resource k in activity i under mode j
nr_{ij}^{l}	the usage of non-renewable resource l in activity i under mode j
$S=(S_0,\ldots,S_{n+1})$	the solution schedule
$S.ET$	the end time of the schedule S
$S_i.act$	the activity corresponds to S_i
$S_i.mode$	the mode corresponds to S_i
$S_i.st$	the start time of S_i, i=0,…,n+1, $0=S_0.st=S_1.st\le S_2.st\le\ldots\le S_{n+1}.st=S.ET$
$S_i.et$	the end time of S_i, i=0,…,n+1, $S_0.st=S_0.et=0$, $S_{n+1}.st=S_{n+1}.et=S.ET$
TE_i	the time point when event i takes place, i=1,…,$\lvert E\rvert$
α	discount rate
$[T_{\mathrm{LOW}},T_{\mathrm{UP}}]$	the expected finish time period, if $S.ET<T_{\mathrm{LOW}}$, bonus will be received. Otherwise, if $S.ET>T_{\mathrm{UP}}$, the payment will be reduced because of the penalty
$I_{[t_i,t_i+1]}$	the indirect cost of the project during the time period unit $[t_i,t_i+1]$
λ and θ	the prepayment rate λ and the milestone payment rate θ
γ and δ	the bonus rate γ and the penalty rate δ
τ_{ij} and η_{ij}	the pheromone value and the heuristic information for the move from $mode_i$ to $mode_j$

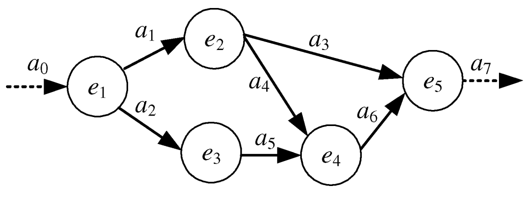


Fig. 1. Example of the AoA network with five events and eight activities (six real activities).

resource l, and its constant expense is CE_{ij}. Different execution modes for a_i may consume different levels of time, cost, and resources. In this sense, there are time–cost, time—resource, and resource–resource tradeoffs between different modes.

4) *Nonpreemption*—Once an activity a_i is executed in mode m_{ij} ($m_{ij}\in M_i$), it has to be completed without any interruption.
5) *Cash flow analysis*—The NPV of the project is analyzed. The objective function of the problem is to maximize the final NPV.

In addition, to make the model more comprehensive and realistic, two more characteristics are taken into account in our formulation.

1) *Indirect costs* (including overhead expenses, finder's fee, etc.) are considered. According to He and Xu [38], indirect costs have been experientially proven to occupy a significant proportion of cash outflows in a project.
2) *A B-P scheme* is incorporated. The B-P scheme has been widely used in contemporary contracting projects [38]. In the B-P scheme, the expected finish time of the project is given by an interval $[t_{\mathrm{LOW}}, t_{\mathrm{UP}}]$. If the project finishes earlier than t_{LOW}, additional rewards will be given. Otherwise, if the finish time is later than t_{UP}, the company will be punished.

The formulation will be presented in detail in the following part of this section. The formulation follows the payment at event occurrences (PEOs) [11] scheme, in which the payments occur at milestone events. Nevertheless, the proposed ACO algorithm can also be applied to the MRCPSPDCF model with other payment schemes.

B. Formulation

The goal of the MRCPSPDCF is to find an optimal schedule $S^* = (S_0^*, \ldots, S_{n+1}^*)$ that maximizes the final NPV. The NPV of a schedule S is evaluated by the following equation.

$$NPV(S) = f_{\text{IN}}(S) - f_{\text{OUT}}(S) + f_{\text{BP}}(S). \tag{1}$$

Here, $NPV(S)$ is the final NPV of schedule S, $f_{\text{IN}}(S), f_{\text{OUT}}(S)$, and $f_{\text{BP}}(S)$ indicate the cash inflows, cash outflows, and bonus or penalty, respectively. Schedule S must satisfy the precedence and resource constraints.

1) Cash Inflows (f_{IN}): f_{IN} consists of the payments at all milestone events. The payment occurs at milestone event e_i is denoted as pay_i. The value of pay_i is discounted to the project-start-time value and the discounted value is denoted as $dpay_i$. f_{IN} is evaluated by using (2)–(6). Equation (2) shows that f_{IN} is composed of three parts: the payment occurs at the beginning of the project ($dpay_1$), at the end of the project ($dpay_{|E|}$), and the payments in midway milestone events

$$\begin{aligned} f_{\text{IN}}(S) &= \sum_{i=1}^{|E|} dpay_i(S) \\ &= dpay_1(S) + \sum_{i=2}^{|E|-1} dpay_i(S) + dpay_{|E|}(S). \end{aligned} \tag{2}$$

At the beginning of the project, a proportion (λ) of the total value (U) is prepaid.

$$dpay_1(S) = pay_1(S) = \lambda U. \tag{3}$$

Suppose that the latest milestone event before e_i occurs at $TLATEST_i$ and e_i occurs at TE_i, the midway payments are evaluated by (4), as shown at the bottom of this page.
Here, θ is the milestone payment rate. pay_i is discounted to $dpay_i$ according to the following equation:

$$dpay_i(S) = pay_i(S) \cdot \exp(-\alpha \cdot TE_i), \qquad 1 < i < |E|. \tag{5}$$

Here, α is the discounted rate. The discounted value of the final payment is given by

$$dpay_{|E|}(S)=[U - pay_1(S) - \sum_{i=2}^{|E|-1} pay_i(S)] \cdot \exp(-\alpha \cdot S.ET). \tag{6}$$

2) Cash Outflows (f_{OUT}): f_{OUT} consists of all expenses during the course of the project, including direct cost (*dcost*) and indirect cost (*icost*)

$$f_{\text{OUT}}(S) = dcost(S) + icost(S). \tag{7}$$

Direct cost *dcost* includes both the constant expenses of activities and the costs of resources. For an execution mode m_{jl}, the resource cost (RC_{jl}) of m_{jl} in each time unit is given by

$$RC_{jl} = \sum_{k=1}^{|RR|} \mathrm{rr}_{jl}^k \cdot PRR_k + \sum_{k=1}^{|NR|} \mathrm{nr}_{jl}^k \cdot PNR_k. \tag{8}$$

The discounted direct cost (DC_{jl}) associated with m_{jl} can be calculated by

$$DC_{jl} = CE_{jl} + \sum_{t'=0}^{d_{jl}-1} RC_{jl} \cdot \exp(-\alpha \cdot t'). \tag{9}$$

Based on the discounted cost of each execution mode, the total direct cost of a schedule S is given by

$$dcost(S) = \sum_{i=1}^{n} DC_{(S_i.act)(S_i.mode)} \cdot \exp(-\alpha \cdot S_i.st). \tag{10}$$

The indirect cost (*icost*) is evaluated by

$$icost(S) = \sum_{t'=0}^{S.ET-1} I_{[t',t'+1]} \cdot \exp[-\alpha(t'+1)]. \tag{11}$$

In a special case, if the indirect costs of all time units equal to a constant value I, according to the summation rule of geometric progression, (11) can be simplified to (12) as follows:

$$icost(S) = \frac{1 - \exp(-\alpha \cdot S.ET)}{\exp(\alpha) - 1} I. \tag{12}$$

3) Bonus and Penalty Analysis (f_{BP}): The B-P value f_{BP} depends on the finish time of the project (*S.ET*). An expected finish period $[T_{\text{LOW}}, T_{\text{UP}}]$ is given. If *S.ET* is earlier than T_{LOW}, the company will gain additional income. Otherwise, if *S.ET* comes later than T_{UP}, the company will lose money due to the penalty of delay. The value of the bonus or penalty should be discounted to the project-start-time value. f_{BP} is evaluated by (13). γ and δ are the bonus and penalty rates, respectively (13), as shown at the bottom of the next page.

III. ACS–MRCPSPDCF Algorithm

The application of ACO requires setting up a construction graph and designing the pheromones and heuristic information. Based on the construction graph, the SSGS [36] is applied for artificial ants to build solutions. In the process of this algorithm, each ant maintains a schedule generator and builds its solution following the rules of ACS using pheromones and heuristic information.

$$\mathrm{dpay}_i(S) = \begin{cases} \theta \sum_{j=1}^{n} isFinish(j, [TLATEST_i, TE_i]) \cdot W_{S_j.act}, & \text{if } e_i \in PE \\ 0, & \text{if } e_i \notin PE \end{cases}, \quad 1 < i < |E|$$

$$\text{where } isFinish(j, [TLATEST_i, TE_i]) = \begin{cases} 1, & \text{if } TLATEST_i < S_j.et \le TE_i \\ 0, & \text{otherwise} \end{cases}. \tag{4}$$

```
procedure conversion
//reorder the index of all modes
   k=0;
   for i=0 to n+1
      for j=1 to |M_i|
         k=k+1;
         mode_k=m_ij;
         Act(mode_k)=i;
         Mode(mode_k)=j;
      end for
   end for
   |MODE|=k;
end procedure
```

Fig. 2. Pseudocode of how to reorder the indexes of modes and generate the set *MODE*.

A. Construction Graph

1) Definition 1—TheMoN Graph: The MoN graph $G = (MODE, EDGE)$ is a complete directed graph. The set of nodes *MODE* is composed of the execution modes of all activities, i.e., $MODE = \bigcup_{i=0}^{n+1} M_i$. The set of directed edges *EDGE* fully connects every two nodes of the graph from both directions.

The AoA network in MRCPSPDCF can be converted into its corresponding MoN graph through the following steps. First, we gather the modes of all activities in a set $MODE = \bigcup_{i=0}^{n+1} M_i = \{mode_1, mode_2, \ldots, mode_{|MODE|}\}$. Then the indexes of all modes are reordered. Each mode m_{ij} corresponds to an element $mode_k$ in the set *MODE*. Fig. 2 shows the pseudocode of how to reorder the indexes. |*MODE*| represents the total number of modes. $mode_1$ is the beginning mode and $mode_{|MODE|}$ is the ending mode. All the other modes are sequentially indexed from $mode_2$ to $mode_{|MODE|-1}$. For the sake of the conversion between these two notations, if $mode_k = m_{ij}$, we denote that $Act(mode_k) = i$ and $Mode(mode_k) = j$.

The set *MODE* becomes the node set of the MoN graph. The set of directed edges fully connects all the nodes from both directions. The edge from $mode_i$ *to* $mode_j$ is denoted as *edge*(i, j). An example of the AoA network and its corresponding MoN graph is given in Fig. 3.

There are two important notes about the MoN graph. First, the MoN graph is the construction graph for ants to build their solutions to the problem. The directed edges become the carriers of pheromones and heuristic information. The pheromone value and the heuristic information on *edge*(i, j) are denoted as τ_{ij} and η_{ij}. Each ant builds a complete solution by choosing these edges step by step according to pheromones and heuristic information. From this point of view, the scheduling problem is converted into a graph-based search problem that aims at finding a path from the beginning node ($mode_1$) to the ending node ($mode_{|MODE|}$) with the maximum NPV, satisfying the precedence and resource constraints. Fig. 3(b) shows a feasible path for the problem. Second, in the MoN graph, there are a lot of redundant directed edges that disobey the precedence constraints and will never be used in any feasible paths. To prevent from processing such redundant edges, each ant maintains an eligible set of edges. Only the edges that satisfy precedence constraints can be selected into the eligible set.

B. Schedule Generator

Based on the construction graph, every single ant maintains a SSGS to build solutions. A solution $S(S_0, \ldots, S_{n+1})$ generated by SSGS is a path from the beginning node ($S_0 =$ *mode*$_1$) to the ending node ($S_{n+1} =$ *mode*$_{|MODE|}$), visiting each activity exactly once. The pseudocode of the SSGS is shown by Fig. 4.

In the initial phase of the SSGS (lines 1–3 in Fig. 4), the beginning node ($mode_1$) is chosen to be the first mode in S. The start time ($S_0.st$) and the end time ($S_0.et$) of *mode*$_1$ are set to 0. Moreover, the ant maintains a set of feasible modes that can be chosen as the next mode in the solution. We denote this set as *eligibleSet*. At this stage, $\forall a_i \in succ(e_1)$, all execution modes of a_i are added to *eligibleSet*.

In the lines 4–9 of the pseudocode, the ant builds a complete solution by iteratively adding eligible modes from *eligibleSet* to the solution S until all activities are covered by S exactly once. In step k, the ant first selects a feasible mode $S_k =$ *mode*$_l$ from *eligibleSet* (line 5) and schedules S_k at the earliest time that satisfies both the precedence and resource constraints (line 6). The function *select*() in line 5 is based on the pseudoproportion random selection rule in the ACS algorithm [32], which will be discussed in the next part of this section. The amount of available resources after executing *mode*$_l$ is updated in line 7. *eligibleSet* is also updated in line 8 by eliminating all the modes that belong to the same activity as *mode*$_l$. Moreover, if the activity to which *mode*$_l$ belongs is one of the direct predecessor activities of event e_j, i.e., $Act(mode_l) \in pred(e_j)$, and all the other activities in $pred(e_j)$ have been chosen to execute, the modes of all activities belonging to $succ(e_j)$ are added to *eligibleSet*. In a special case, if nonrenewable resources are not enough for the execution of any unexecuted modes, the schedule generator will be terminated and the NPV of this infeasible schedule will be set to a very small value.

After constructing a complete schedule S, the NPV of S is evaluated based on the equations given in Section III.

C. ACS–MRCPSPDCF Algorithm

The flowchart of the ACS–MRCPSPDCF algorithm is given in Fig. 5. In the algorithm, a group of ants is dispatched to build solutions iteratively. Each ant maintains a single SSGS to build

$$f_{\mathrm{BP}}(S) = \begin{cases} \gamma U(T_{\mathrm{LOW}} - S.ET) \cdot \exp(-\alpha \cdot S.ET), & \text{if } S.ET < T_{\mathrm{LOW}} \\ 0, & \text{if } T_{\mathrm{LOW}} \le S.ET \le T_{\mathrm{UP}} \\ -\delta U(S.ET - T_{\mathrm{UP}}) \cdot \exp(-\alpha \cdot S.ET), & \text{if } T_{\mathrm{UP}} < S.ET \end{cases} \quad (13)$$

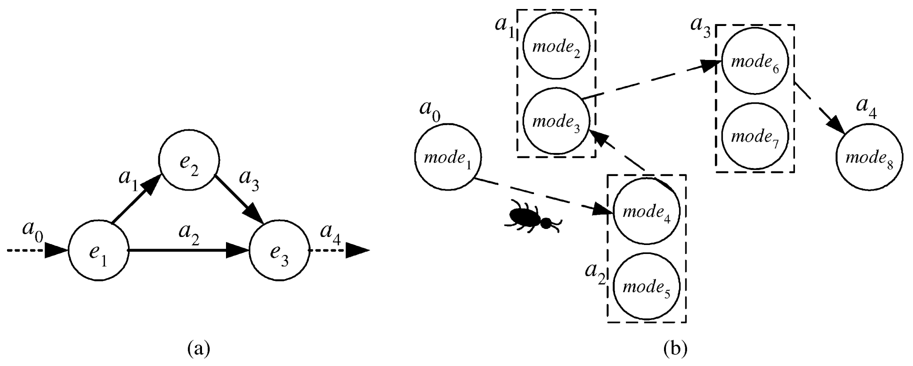


Fig. 3. Example of the conversion from an AoA network to its corresponding MoN graph. (a) AoA network with three events, three real activities, and two dummy activities (a_0 and a_4). Assume that each real activity has two modes. In all, there are eight modes, which are shown in (b). The eight nodes in (b) and the set of directed edges that fully connects all the nodes from both directions compose the MoN graph, which is the construction graph for the ACS algorithm. (b) Example of a feasible solution path.

```
Procedure SSGS
1    $S_0 = mode_1$;
2    $S_0.st = S_0.et = 0$;
3    Update eligibleSet;
4    for i=1 to n+1
5        $S_i$=select(); // select a mode from eligible set
6        Schedule $S_i$ to the earliest time when the precedence
             and resource constraints are satisfied;
7        Update Resources;
8        Update eligibleSet;
9    end for
10   evaluate NPV(S);
end procedure
```

Fig. 4. Pseudocode of SSGS.

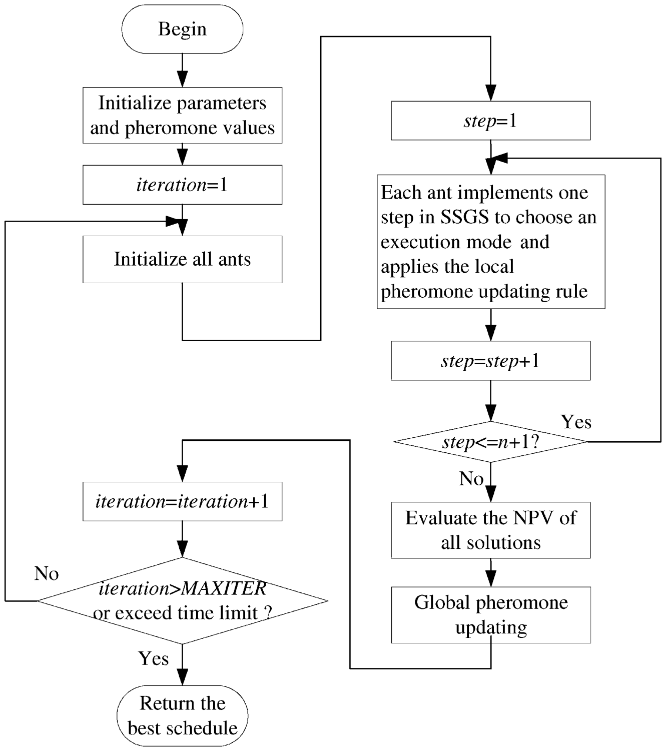
its own solution in every iteration. The ants also maintain the pheromone values and heuristic information and use them in the selection scheme of the SSGS.

1) Selection Scheme: The next mode is chosen by the function *select*( ) in the SSGS (line 5 in Fig. 4) according to the pseudo proportion random selection rule. Suppose an ant is located at $mode_i$. The next mode $mode_j$ returned by the function *select*() is determined by the following equations:

$$mode_j = \begin{cases} \arg\max_{mode_l \in eligibleSet}\{\tau_{il}\cdot(\eta_{il})^\beta\}, & \text{if } q \le q_0 \\ \text{implement the roulette wheel scheme,} & \text{otherwise} \end{cases} \tag{14}$$

$$p(mode_j) = \begin{cases} \dfrac{\tau_{ij}\cdot(\eta_{ij})^\beta}{\sum_{mode_l \in eligibleSet}\tau_{il}\cdot(\eta_{il})^\beta}, & \text{if } mode_j \in eligibleSet \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

A random number $q \in [0, 1]$ is generated and compared with a parameter q_0. If $q \le q_0$, the mode $mode_l$ that has the largest value of $\tau_{il}\cdot(\eta_{il})^\beta$ among all the modes in *eligibleSet* is selected. τ_{il} and η_{il} are respectively the pheromone value and heuristic information on the directed edge *edge*(i,l) from $mode_i$ to $mode_l$. β is a parameter that determines the weight of heuristic information. Otherwise, if $q > q_0$, the roulette wheel scheme is adopted, i.e., the probability of selecting $mode_j$ is in direct proportion to the value of $\tau_{ij}\cdot(\eta_{ij})^\beta$, if $mode_j \in eligibleSet$.



Begin
Initialize parameters and pheromone values
iteration=1
Initialize all ants
step=1
Each ant implements one step in SSGS to choose an execution mode and applies the local pheromone updating rule
step=step+1
step<=n+1?
Yes
No
Evaluate the NPV of all solutions
Global pheromone updating
iteration=iteration+1
iteration>MAXITER or exceed time limit ?
No
Yes
Return the best schedule

Fig. 5. Flowchart of the ACS–MRCPSPDCF algorithm.

2) Pheromone Management: At the beginning of the ACS–MRCPSPDCF algorithm, the pheromone values on all directed edges are initialized to $\tau_0 = 1$.

$$\tau_{ij} = \tau_0 = 1 \qquad \forall(i, j) \in EDGE \tag{16}$$

Here, τ_0 is also the lower bound of all pheromone values, i.e., the pheromone management procedure guarantees $\tau_{ij} \ge \tau_0$ $\forall(i, j) \in EDGE$.

Immediately after a mode is selected by an ant, the local pheromone updating procedure is performed. Assuming that an ant located at $mode_i$ chooses $mode_j$ to move, the local pheromone updating rule is given by

$$\tau_{ij} = (1-\xi)\tau_{ij} + \xi\tau_0 \tag{17}$$

where $\xi \in [0,1]$ is a parameter. Because τ_0 is the lower bound of all the pheromone values, in general, we have $\tau_0 \le (1-\xi)\tau_{ij} + \xi\tau_0 \le \tau_{ij}$. Therefore, the effect of (17) is actually to decrease the value of τ_{ij}, if $\tau_{ij} > \tau_0$. In other words, the local pheromone updating rule decreases the probabilities for the following ants to choose the same edge (i, j). In this case, the following ants have higher probabilities to explore other unused edges, and thus the algorithm is able to maintain diversity to explore different solutions.

After all ants have finished constructing their solutions at the end of each iteration, the global pheromone updating rule is used to reinforce the best schedule. Only the edges on the best-so-far schedule are able to obtained additional pheromones based on the following rule:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta_{ij}^{bs} \tag{18}$$

where $\rho \in [0,1]$ is a parameter and Δ_{ij}^{bs} is defined based on the NPV of the best-so-far schedule S^{bs} as follows:

$$\Delta_{ij}^{bs} = \left(\frac{NPV(S^{bs}) - \mathit{first}}{\mathit{firstBest} - \mathit{first}} + 1\right) \cdot \psi. \tag{19}$$

In (19), *first* is the NPV of the first feasible solution found since the beginning of the algorithm and *firstBest* is the NPV of the first feasible iteration-best solution whose value is larger than*first*. $\psi \ge 1$ is a parameter to control the scale of pheromone amount. If no feasible solutions have ever been found by the algorithm (*first* is not defined), the global pheromone updating rule is not applied. If feasible solutions have been found but the value of *firstBest* has not been defined, we set $\Delta_{ij}^{bs} = \psi$.

According to [35] and [39], if all the pheromone values are limited in an interval $[\tau_{\min}, \tau_{\max}]$ ($\tau_{\min} > 0$ and $\tau_{\max} < +\infty$), the convergence of the ACS algorithm can be guaranteed. In the previous pheromone management scheme, both the local and the global pheromone updating rules have the same format

$$a_{k+1} = (1-\theta)a_k + \theta b \tag{20}$$

where a_{k+1} is the pheromone value after updating, a_k is the pheromone value before updating, $\theta = \xi$ or ρ, $k = 1, 2, \ldots$, and $b = \tau_0$ or Δ_{ij}^{bs}. Based on (16) and (19), we have $\tau_0 = 1$ and

$$1 \le \Delta_{ij}^{bs} \le \left(\frac{NPV(S^*) - first}{firstBest - first} + 1\right) \cdot \psi \tag{21}$$

where $\mathrm{NPV}(S^*)$ means the NPV value of the global optimal schedule S^*. The right-hand side item in (21) is a function of the global optimal schedule S^*. We denote it as

$$g(S^*) = \left(\frac{NPV(S^*) - \mathit{first}}{\mathit{firstBest} - \mathit{first}} + 1\right) \cdot \psi. \tag{22}$$

Therefore, we have $1 \le b \le g(S^*)$.

Furthermore, according to (20), we can deduce that

$$\begin{aligned} a_1 &= (1-\theta)a_0 + \theta b \\ a_2 &= (1-\theta)a_1 + \theta b \\ &= (1-\theta)[(1-\theta)a_0 + \theta b] + \theta b \\ &= (1-\theta)^2 a_0 + [1-(1-\theta)^2]b \\ &\ldots \\ a_k &= (1-\theta)^k a_0 + [1-(1-\theta)^k]b. \end{aligned} \tag{23}$$

Because $a_0 = \tau_0 = 1$, $\theta \in [0,1]$, and $1 \le b \le g(S^*)\ \forall k = 1, 2, \ldots$, it can be deduced that

$$\begin{aligned} a_k = (1-\theta)^k a_0 + [1-(1-\theta)^k]b &\ge (1-\theta)^k a_0 \\ &+ [1-(1-\theta)^k]a_0 = a_0 = 1 \end{aligned} \tag{24}$$

and

$$\begin{aligned} a_k = (1-\theta)^k a_0 + [1-(1-\theta)^k]b &\le (1-\theta)^k b \\ &+ [1-(1-\theta)^k]b = b \le g(S^*). \end{aligned} \tag{25}$$

In other words, all the pheromone values are limited to the interval $[1, g(S^*)]$. So the convergence of the proposed ACS–MRCPSPDCF algorithm can be guaranteed.

3) Heuristic Information: In this paper, eight different heuristics are used to enhance the search ability of ants, including two precedence-relation-based heuristics, two time-based heuristics, two cost-based heuristics, a resource-based heuristic, and a hybrid heuristic. As there are time–cost, time—resource, and resource–resource tradeoffs in the MRCPSPDCF, these eight heuristics use the information from different aspects of the tradeoffs to guide the search direction of ants.

a) Maximum total successors: Maximum total successors (MTS) is a precedence-relation-based heuristic designed for the classical RCPSP [18]. Suppose $mode_j$ is an alternative execution mode for activity a_k and the number of all successor activities of a_k is $allson_k$, the heuristic η_{ij} is set to $\eta_{ij} = allson_k$. The aim of the MTS heuristic is to first schedule the activities with the largest number of total successors.

b) Weight resource utilization and precedence: Weight resource utilization and precedence (WRUP) is also a precedence-relation-based heuristic [18], [40]. If $mode_j$ is the lth alternative execution mode for activity a_k, the heuristic η_{ij} is evaluated as follows:

$$\eta_{ij} = 0.5 \times |succ(a_k)| + 0.5 \times \sum_{h=1}^{|RR|} \frac{rr_{kl}^h}{RR_h}. \tag{26}$$

Here, $|succ(a_k)|$ is the number of direct successor activities of a_k. The effect of the WRUP heuristic is to expedite the bottleneck activities that have many direct successors and take up a lot of resources.

c) Earliest finish time: Earliest finish time (EFT) is a time-based heuristic for the classical RCPSP [18]. Given a precedence network, the earliest start time (EST) of each activity a_k is determined by the critical path of the network [18]. We denote the EST of a_k as EST_k. Suppose $mode_j$ is the lth alternative execution mode for activity a_k the EFT of *mode*$_j$ is given by

$$EFT_j = EST_k + d_{kl}. \tag{27}$$

Then, the heuristic η_{ij} is evaluated by

$$\eta_{ij} = \frac{1}{EFT_j}. \tag{28}$$

d) Shortest processing time: Shortest processing time (SPT) is a time-based heuristic for MRCPSP [41]. Suppose *mode*$_j$ is the lth alternative execution mode for activity a_k, the heuristic η_{ij} is evaluated by

$$\eta_j = \frac{1}{d_{kl}}. \tag{29}$$

The SPT heuristic biases the modes with shorter execution time.

e) Minimal cost: Minimal cost (MC) is a cost-based heuristic that considers the total cost of each mode. Suppose *mode*$_j$ is the lth alternative execution mode for activity a_k, the heuristic η_{ij} is given by

$$\eta_{ij} = \frac{1}{CE_{kl} + (RC_{kl} + I) \cdot d_{kl}} \tag{30}$$

where CE$_{kl}$ and RC$_{kl}$ are the constant expenses and resource costs of m_{kl}, respectively.

f) Minimal discounted cost: To satisfy the need of the MRCPSPDCF, we take the time value of money into account and propose the minimal discounted cost (MDC) heuristic. Suppose *mode*$_j$ is the lth alternative execution mode for activity a_k, the heuristic η_{ij} is evaluated by

$$\eta_{ij} = \frac{1}{CE_{kl} + \sum_{t'=0}^{d_{kl}-1} (RC_{kl} + I) \cdot \exp(-\alpha \cdot t')}. \tag{31}$$

The MDC heuristic biases the execution modes that cost less money with respect to the discounted values.

g) Minimal resource consumed: To consider the aspect of resources, we also design the minimal resource consumed (MRC) heuristic for the proposed ACO algorithm. Suppose *mode*$_j$ is the lth alternative execution mode for activity a_k, the heuristic η_{ij} is evaluated by

$$\eta_{ij} = \frac{1}{d_{kl}\left(\sum_{h=1}^{|RR|} rr_{kl}^h / RR_h + \sum_{h=1}^{|NR|} nr_{kl}^h / NR_h\right)}. \tag{32}$$

The MRC heuristic biases the modes that use fewer resources.

h) Hybrid heuristic: The hybrid heuristic aims to combine the information from all aspects. Four of the aforementioned heuristics are selected in the hybrid heuristic, including MTS, EFT, MDC, and MRC. These four heuristics belong to four different types. In the hybrid heuristic, in each step of SSGS, before calling the function *select*( ), the ants randomly choose one of the previous four heuristics to build new solutions.

Note that the MTS, WRUP, EFT, SPT, MC, MDC, and MRC heuristics are all static heuristics, i.e., the values of these heuristics are only determined by the instance. Therefore, in the implementation of the algorithm, these heuristics only need to be evaluated once at the beginning, and remain unchanged during the whole search process. In this way, the management of these heuristics is not time-consuming.

IV. Experimental Results

A. Test Instances and Parameter Settings

A set of 55 random instances are used to study the performance of the proposed ACS–MRCPSPDCF algorithm. The instances are of eleven different sizes (numbers of activities), i.e., $n = \{13, 16, 18, 28, 36, 40, 48, 58, 60, 80, 98\}$. Each size corresponds to five instances. We name the five instances with 13 activities as Ins13_1, Ins13_2, . . ., Ins13_5. The other instances are named in a similar way. The terms "instance 1" to "instances 5" in Table II correspond to the five instances of each size, respectively. In the instances, each activity is randomly associated with one to five different execution modes. Resources and cash flows are all randomly generated.

We first set the parameters of the ACS–MRCPSPDCF algorithm: the number of ants used in each iteration is 10, $\zeta = 0.1$, $\rho = 0.1$, $\beta = 1, q_0 = 0.9$, and $\psi = 20$. The first three parameters are set according to the original ACS algorithm for traveling salesman problem (TSP) [32]. These configurations still contribute to good performance with respect to the MRCPSPDCF. For the other three parameters, experimental results suggest that the aforementioned configuration manages to obtain promising performance.

The parameter β in (14) and (15) is used to weight the importance of pheromones and heuristics. If $\beta = 0$, the algorithm does not use any heuristic information but only the pheromone values are at work. In contrast, a relative large β reinforces the influence of heuristic information. But if β is too large, the algorithm tends to behave in a greedy-search manner and becomes easily trapped in local optima. Fig. 6 illustrates the performance of the algorithm when β is set to 0, 1, 2, 3, 5, and 8, respectively. In the figure, the label "evaluations" means the maximum number of NPV evaluations during the process of the algorithm. As the evaluation of NPV is the most time-consuming part of the algorithm, Fig. 6 can be used to compare the search speed of the algorithm with different β values. In the experiment, it is found that the performance of the algorithm with no heuristics ($\beta = 0$) is rather poor and the NPV values obtained by this setting are always out of scales of the plots. Thus, the curves for $\beta = 0$ are not shown in Fig. 6. When β is large, the algorithm stagnates in the early stage and the performance is not good either. Overall, $\beta = 1$ is able to balance the usage of pheromones and heuristics and performs well in most cases.

The parameter q_0 in (18) is used to balance the exploration and exploitation behaviors of the algorithm. In (18), a random number $q \in [0, 1]$ is generated. If $q \leq q_0$, the ant directly chooses the best mode according to the pheromone and heuristic values. In this case, the ant is exploiting the learned knowledge and thus this selection scheme is called exploitation [32], [35]. On the other hand, if $q > q_0$, the ant uses the roulette wheel selection scheme. In this case, the ant has a higher probability to choose the modes that have never been explored in advance. Thus, this selection scheme is called exploration [32], [35]. The performance of the algorithm with different q_0 values is

TABLE II
COMPARISON BETWEEN THE ACS–MRCPSPDCF AND THE PGA, HGA, SA, AND TS APPROACHES ON 55 INSTANCES
(ALL RESULTS ARE AVERAGED OVER 50 INDEPENDENT RUNS)

size	Algorithm	Number of Evaluations	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
13	ACS	10000	**-154.58**	**1781.69**	**2713.87**	2254.23	**14609.9**
	PGA	14000	-269.17	1676.33	2683.02	2333.36	14494.8
	HGA	14000	-456.056	1765.24	2695.03	**2359.98**	14445
	SA	21000	-1755.23	-57.046	1039.31	1500.35	13755.4
	TS	21000	-1731.86	356.724	755.752	1359.04	13559.0
	ACS-PGA (*t*-test)		1.6958	1.7843	0.4349	-1.4301	**3.8163#**
	ACS-HGA (*t*-test)		**3.5722#**	0.3798	0.2793	-2.0234#	**6.5372#**
	ACS-SA (*t*-test)		**26.2521#**	**24.9945#**	**18.8286#**	**13.6609#**	**17.2165#**
	ACS-TS (*t*-test)		**24.6035#**	**18.8830#**	**19.5471#**	**14.0738#**	**20.7727#**
16	ACS	10000	**6939.70**	**-1107.40**	**15626.7**	2654.47	14484.8
	PGA	14000	6887.86	-1216.37	15548.7	**2780.53**	14458.2
	HGA	14000	6628.85	-1140.85	15397.6	2638.55	**14567.4**
	SA	21000	5305.55	-2540.96	14593.7	2301.65	13479.7
	TS	21000	5325.84	-2416.17	14681.5	1973.54	13523.1
	ACS-PGA (*t*-test)		**2.2588#**	**3.1866#**	**2.7612#**	-4.4096#	0.3632
	ACS-HGA (*t*-test)		**12.6020#**	**2.7479#**	**8.6070#**	0.5475	-1.3545
	ACS-SA (*t*-test)		**25.2549#**	**16.2415#**	**21.4747#**	**6.0317#**	**15.2520#**
	ACS-TS (*t*-test)		**24.8439#**	**15.1814#**	**20.1334#**	**16.3290#**	**14.9850#**
18	ACS	10000	1498.14	12592.9	**7293.96**	**3688.31**	**7633.23**
	PGA	14000	**1738.35**	**12622.8**	6915.91	3688.29	7475.26
	HGA	14000	1578.05	12465.1	7082.78	3680.69	7500.5
	SA	21000	806.152	10256.5	4877.71	3469.77	7133.83
	TS	21000	462.17	10202.5	4971.08	3547.04	6774.62
	ACS-PGA (*t*-test)		-2.4426#	-0.5579	**5.2029#**	0.011	**4.9451#**
	ACS-HGA (*t*-test)		-0.9710	1.9723	1.9183	0.378	**3.7692#**
	ACS-SA (*t*-test)		**6.4465#**	**22.5317#**	**30.9946#**	**16.7874#**	**5.9216#**
	ACS-TS (*t*-test)		**15.4713#**	**24.7215#**	**29.3634#**	**12.2487#**	**12.4663#**
28	ACS	20000	**814.287**	7579.83	**976.407**	**-329.217**	**4153.34**
	PGA	28000	589.408	**7642.26**	696.312	-329.633	4130.77
	HGA	28000	802.313	7522.05	717.277	-361.608	3830.13
	SA	42000	181.425	5069.23	9.51412	-1844.74	-484.267
	TS	42000	-3072.86	4975.25	-2807.39	-1785.28	-220.486
	ACS-PGA (*t*-test)		**2.6687#**	-1.8680	**7.4018#**	0.0189	0.7769
	ACS-HGA (*t*-test)		0.1391	1.6045	**6.0112#**	1.3352	**8.5759#**
	ACS-SA (*t*-test)		**3.0821#**	**44.1782#**	**8.2658#**	**43.6541#**	**48.4618#**
	ACS-TS (*t*-test)		**33.8282#**	**46.8470#**	**34.0744#**	**38.3750#**	**43.8274#**
36	ACS	30000	**2548.91**	**10304.8**	**746.978**	**11288.4**	**-2015.85**
	PGA	42000	2399.53	10184.3	376.165	11222.8	-2353.74
	HGA	42000	2291.66	10198.9	594.35	10870.6	-2172.94
	SA	63000	-1538.51	5810.58	-867.955	8923.99	-3557.14
	TS	63000	-1698.94	5769.97	-1826.62	5700.88	-4426.09
	ACS-PGA (*t*-test)		**2.8135#**	**2.5330#**	**8.0266#**	0.8097	**10.3857#**
	ACS-HGA (*t*-test)		**4.2993#**	**2.4476#**	**4.0848#**	**5.2877#**	1.7882
	ACS-SA (*t*-test)		**51.6003#**	**50.5355#**	**7.7473#**	**6.7025#**	**19.8891#**
	ACS-TS (*t*-test)		**66.3235#**	**52.7396#**	**54.1255#**	**30.0745#**	**44.7369#**
40	ACS	30000	**-871.033**	7546.27	**2293.5**	7090.27	**6049.08**
	PGA	42000	-1087.46	**7695.99**	1915.82	6596.31	5084.17
	HGA	42000	-1274.11	7513.6	1915.84	**7303.73**	5755.87
	SA	63000	-5940.98	6476.57	-3890.69	2132.03	1548.92
	TS	63000	-6215.17	6453.37	-6041.81	1922.83	-1082.48
	ACS-PGA (*t*-test)		1.7496	-7.5484#	**2.5268#**	**3.5946#**	**4.6949#**
	ACS-HGA (*t*-test)		**3.2926#**	0.7543	**2.4997#**	-2.0911	**2.4414#**
	ACS-SA (*t*-test)		**35.9335#**	**35.5834#**	**12.1188#**	**29.8251#**	**8.6149#**
	ACS-TS (*t*-test)		**60.3300#**	**41.6050#**	**53.8311#**	**46.6855#**	**63.3837#**
48	ACS	40000	10792.6	**10690.5**	**9840.26**	20898.9	**22161.7**
	PGA	56000	10794.6	10099.1	9672.02	**20984**	21614.4
	HGA	56000	**10836.5**	10298	9358.46	20816.9	21954
	SA	84000	6985.64	3504.99	2643.38	17443.6	16420.3
	TS	84000	6746.74	3527.84	2431.39	17356.5	16283.8
	ACS-PGA (*t*-test)		-0.0429	**4.7416#**	**3.5669#**	-1.2246	**5.7431#**
	ACS-HGA (*t*-test)		-1.0468	**3.0624#**	**10.5221#**	1.1493	1.9761
	ACS-SA (*t*-test)		**65.9458#**	**62.7684#**	**56.4875#**	**25.4352#**	**58.4956#**
	ACS-TS (*t*-test)		**74.2564#**	**61.5503#**	**66.1167#**	**27.2581#**	**64.3076#**

TABLE II
(Continued)

	ACS	50000	**9117.69**	**-1867.39**	9918.87	**9275.19**	**9571.67**
	PGA	70000	6606.23	-2135.71	8994.73	9200.48	9113.19
	HGA	70000	8284.35	-2839.28	**10428.9**	9211.43	9318.88
	SA	105000	325.232	-6516.06	5999.01	5932.06	6876.76
58	TS	105000	-3040.3	-13495.8	30.2424	2830.51	3647.89
	ACS-PGA (*t*-test)		**10.9689#**	**2.89654#**	**3.6469#**	0.8138	**4.7299#**
	ACS-HGA (*t*-test)		**6.8391#**	**9.0169#**	-2.0136	0.6279	**2.6800#**
	ACS-SA (*t*-test)		**10.9226#**	**5.4828#**	**6.4860#**	**6.3798#**	**6.8078#**
	ACS-TS (*t*-test)		**58.5851#**	**74.6570#**	**61.8834#**	**57.6374#**	**55.6707#**
	ACS	50000	**-6008.9**	**5694.65**	**46310.4**	**8902.57**	**14787.9**
	PGA	70000	-6379.32	5489.72	45831.5	8423.06	14276.3
	HGA	70000	-7517.64	4943.25	45855.5	7081.21	12824
	SA	105000	-17024.7	-4447.78	31317.7	-4776.22	-2390.17
60	TS	105000	-16998.9	-5190.32	31364.6	-5088.07	-2288.31
	ACS-PGA (*t*-test)		**6.1869#**	**2.1945#**	**2.0299#**	**4.4884#**	**2.0693#**
	ACS-HGA (*t*-test)		**19.7974#**	**9.1333#**	**2.0265#**	**15.9265#**	**7.3820#**
	ACS-SA (*t*-test)		**94.8118#**	**22.1462#**	**64.6464#**	**43.3987#**	**63.8966#**
	ACS-TS (*t*-test)		**69.3866#**	**94.9799#**	**53.5113#**	**83.7415#**	**60.4529#**
	ACS	80000	**16149.7**	**-476.501**	**9291.46**	**22603.4**	**9598.99**
	PGA	112000	15851.0	-1542.28	8733.97	22550.1	9418.37
	HGA	112000	15377.5	-2174.83	8580.82	21750.1	8193.32
	SA	168000	5634.54	-22115.7	-2471.63	12871.2	-7940.28
80	TS	168000	5347.72	-22270.1	-4022.49	12808	-8280.27
	ACS-PGA (*t*-test)		**3.3367#**	**4.1547#**	**4.8106#**	**2.0779#**	**2.6724#**
	ACS-HGA (*t*-test)		**10.4855#**	**8.2087#**	**6.9625#**	**12.9614#**	**16.2155#**
	ACS-SA (*t*-test)		**86.5509#**	**99.2105#**	**21.9186#**	**105.527#**	**104.475#**
	ACS-TS (*t*-test)		**92.7033#**	**104.641#**	**97.3822#**	**123.370#**	**122.831#**
	ACS	100000	**-3281.68**	**958.735**	**20500.3**	**32321.9**	**21551.1**
	PGA	140000	-4336.94	283.996	20204.9	31178.3	21323.1
	HGA	140000	-4155.68	-1982.45	19657.3	31798.2	19963.9
	SA	210000	-21097.7	-19362.4	-2966.59	26426.8	6143.76
98	TS	210000	-23961.3	-28065.2	-3647.63	24042.7	5929.05
	ACS-PGA (*t*-test)		**4.6617#**	**5.7676#**	**2.9987#**	**11.2952#**	**3.1684#**
	ACS-HGA (*t*-test)		**4.0238#**	**25.4095#**	**8.0218#**	**4.9981#**	**19.8625#**
	ACS-SA (*t*-test)		**19.0892#**	**21.1621#**	**117.462#**	**13.1224#**	**114.032#**
	ACS-TS (*t*-test)		**116.304#**	**143.064#**	**159.261#**	**93.3670#**	**122.683#**

#The value of t with 49 DOFs is significant at α=0.05 by a two-tailed test.

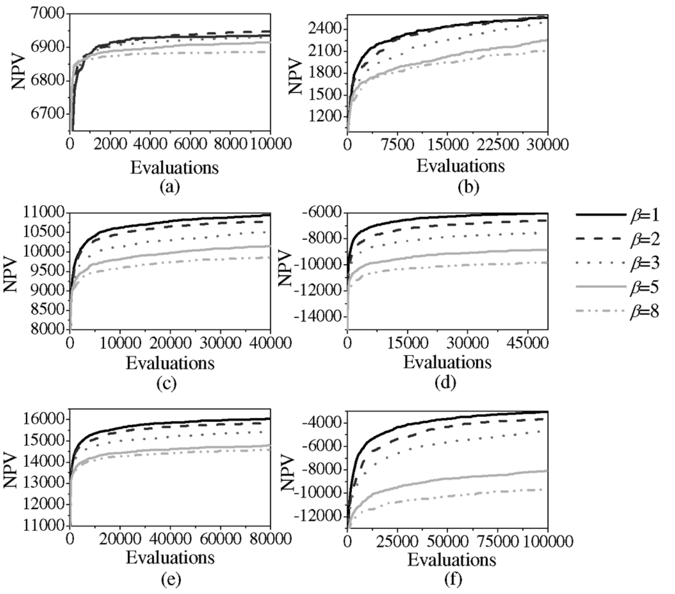


Fig. 6. Performance of the ACS–MRCPSPDCF algorithm with different values for β. The results are averaged over 50 independent runs. (a) Instance Ins16_1 with 16 activities. (b) Instance Ins36_1 with 36 activities. (c) Instance Ins48_1 with 48 activities. (d) Instance Ins60_1 with 60 activities. (e) Instance Ins80_1 with 80 activities. (f) Instance Ins98_1 with 98 activities.

plotted in Fig. 7. In the experiment, the algorithm stops when it reaches the maximum number of NPV evaluations. From the results given in the figure, obviously, $q_0 = 0.9$ is the best choice in most cases. This is because the artificial ants in the algorithm are more likely to exploit the good modes found previously when q_0 is large. However, if $q_0 = 1$, no exploration is made. In this case, the algorithm stagnates in the early stage and the performance is poor. The earlier results show that the function of q_0 in the proposed algorithm is consistent with that in the original ACS algorithm for the TSP [32].

The parameter ψ in (23) is used to control the scale of pheromone amount. If ψ is set to a small value, the differences between pheromone values on all edges become small and the influence of pheromone is reduced. In the experiment, ψ is set to $\{5, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$ and some of the results are given in Fig. 8. Experimental results show that the performance of $\psi = 5, 10,$ and 15 is not good. This is because the differences between pheromone values are too small to reflect the previous search experiences. According to Fig. 8, the algorithms with $\psi \in [20, 50]$ are all able to obtain relatively good results.

B. Performance of Different Heuristics

Eight different heuristics have been considered in the proposed algorithm, including MTS, WRUP, EFT, SPT, MC, MDC,

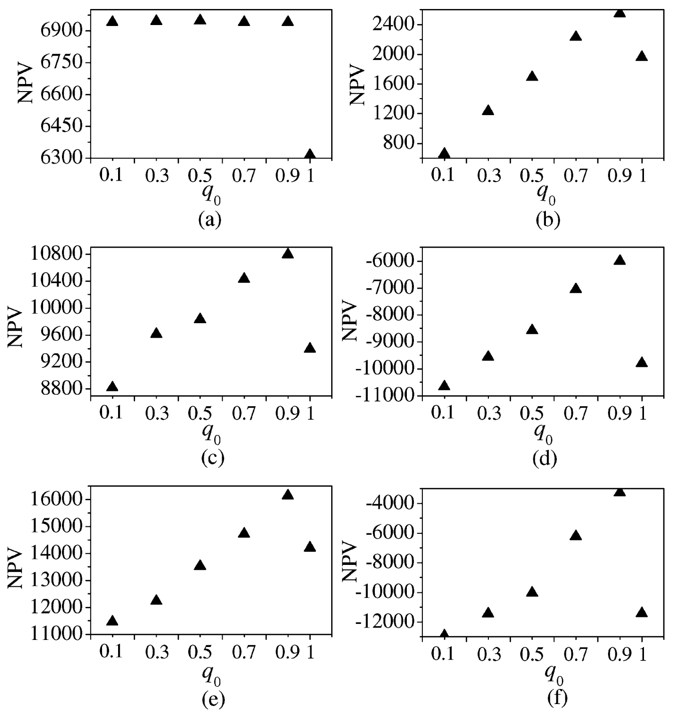


Fig. 7. Performance of the ACS–MRCPSPDCF algorithm with different values for q_0. The results are averaged over 50 independent runs. (a) Instance Ins16_1 with 16 activities. (b) Instance Ins36_1 with 36 activities. (c) Instance Ins48_1 with 48 activities. (d) Instance Ins60_1 with 60 activities. (e) Instance Ins80_1 with 80 activities. (f) Instance Ins98_1 with 98 activities.

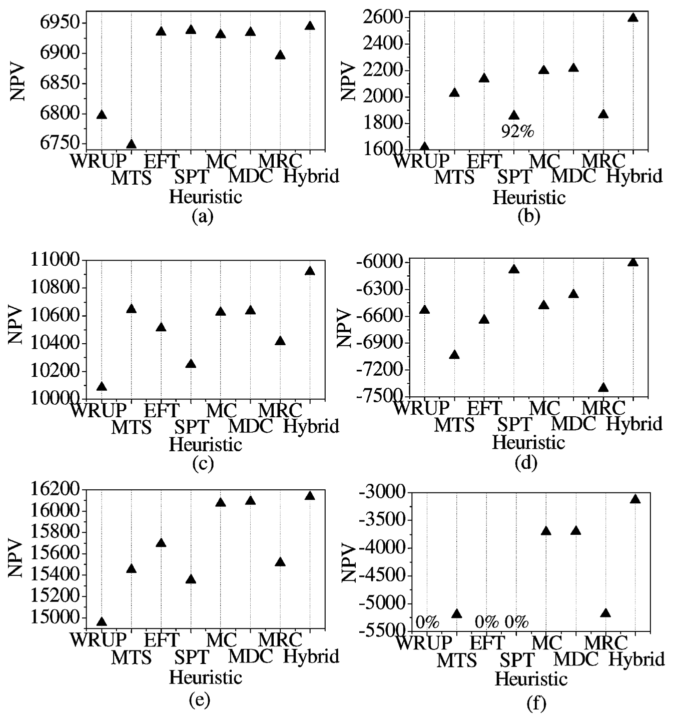


Fig. 9. Performance of the ACS–MRCPSPDCF algorithm with different heuristics. The results are averaged over 50 independent runs. (a) Instance Ins16_1 with 16 activities. (b) Instance Ins36_1 with 36 activities. (c) Instance Ins48_1 with 48 activities. (d) Instance Ins60_1 with 60 activities. (e) Instance Ins80_1 with 80 activities. (f) Instance Ins98_1 with 98 activities.

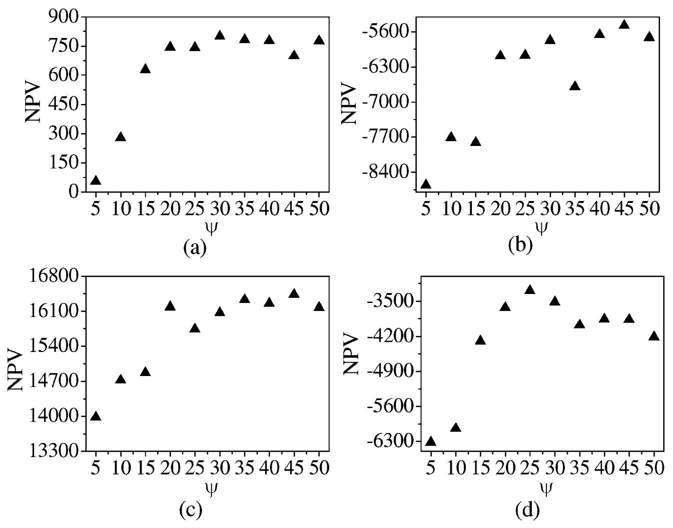


Fig. 8. Performance of the ACS–MRCPSPDCF algorithm with different values for ψ. The results are averaged over 50 independent runs. (a) Instance Ins36_3 with 36 instances. (b) Instance Ins60_1 with 60 instances. (c) Instance Ins80_1 with 80 instances. (d) Instance Ins98_1 with 98 instances.

MRC, and the hybrid heuristic. The performance of the ACS–MRCPSPDCF algorithm with different heuristics is shown in Fig. 9. In the experiment, the algorithm stops when it reaches the maximum number of NPV evaluations. The results marked by triangles in the figure are the averaged results of all the runs where feasible solutions are successfully found. In some difficult instances with tight nonrenewable resource constraints, a heuristic may sometimes fail to find feasible solutions. Therefore, the percentages of successful runs are also given in Fig. 9. If the percentage is not provided, it means that the heuristic is able to find feasible solutions in all runs.

The precedence-relation-based heuristics MTS and WRUP is designed to expedite the critical activities with respect to the precedence relations [18]. If an activity has many successors, the precedence-relation-based heuristics intend to schedule this activity first so that the successors can be processed as soon as possible. According to the results in Fig. 9, the performance of MTS and WRUP is not remarkable compared with the other heuristics. However, for the instances characterized by having sufficient renewable resources for the parallel execution of several different modes (e.g., the instance Ins48_1), the MTS is able to yield promising results [see Fig. 9(c)].

The time-based heuristics such as EFT and SPT are important for the classical RCPSP with makespan criterion [18]. For the MRCPSPDCF, time-based heuristics are still useful as time value of money is considered in the model. Fig. 9 reveals that EFT and SPT are able to yield good results on some instances, for example, the instance Ins16_1 [see Fig. 9(a)] and Ins60_1 [see Fig. 9(d)]. However, as these two heuristics only consider the influence of time, they sometimes fail to find feasible solutions for the instances with tight nonrenewable resource constraints [see Fig. 9(b) and (f)].

Cost-based heuristics are directly related to the cash flow criterion. Thus, the performance of the MC and MDC heuristics is good in all instances. As the MDC heuristic considers the

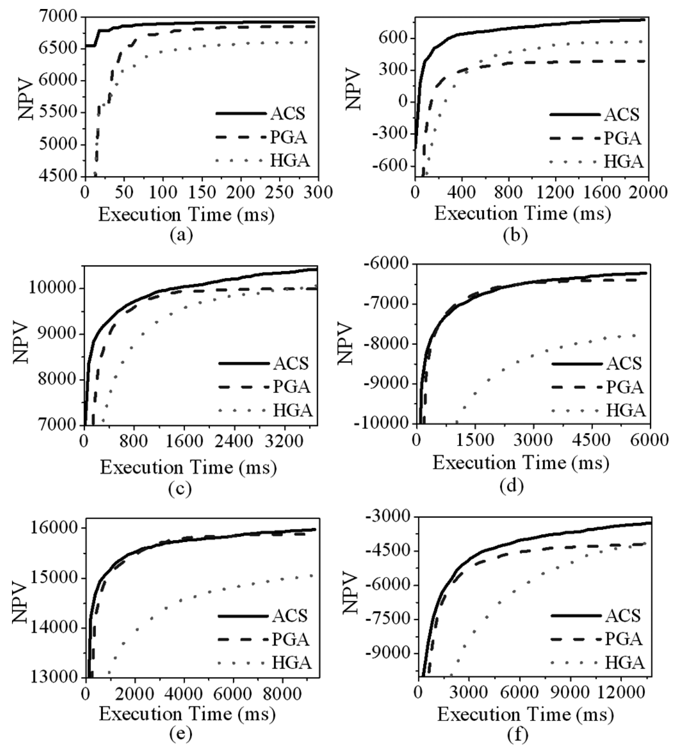


Fig. 10. The search speed of ACS–MRCPSPDCF, PGA, and HGA with respect to the CPU time. All results are averaged over 50 independent runs. (a) Instance Ins16_1 with 16 activities. (b) Instance Ins36_3 with 36 activities. (c) Instance Ins48_2 with 48 activities. (d) Instance Ins60_1 with 60 activities. (e) Instance Ins80_1 with 80 activities. (f) Instance Ins98_1 with 98 activities.

time value of money, its performance is slightly better than MC according to Fig. 9.

For the resource-based heuristic MRC, it is designed to bias the modes that use fewer resources. Although the performance of MRC is not as good as the cost-based heuristics, MRC is able to help artificial ants to build feasible solutions that satisfy the nonrenewable resource constraints.

In addition to the previous heuristics, we also combine four different heuristics to propose a hybrid heuristic. In this way, the hybrid heuristic is able to consider the problem from different angles. According to the results in Fig. 9, the hybrid heuristic significantly outperforms the other heuristics in all instances.

C. Comparison with the Other Existing Algorithms for the MRCPSPDC

In order to demonstrate the effectiveness of the ACS–MRCPSPDCF algorithm, the proposed approach is compared with the pure GA (PGA) proposed by Ulusoy [11], the hybrid GA (HGA) proposed by Özdamar [18], and the SA and TS algorithms proposed by Mika *et al.* [14]. The encoding scheme of the PGA in [11] is based on the mode assignment and the order of activities. A multicomponent uniform order-based crossover operator (MCUOX) and two mutation operators are defined in the algorithm. The parameter configurations of the GA in [11] are as follows: population size $popsize = 140$, crossover probability $pc = 0.65$, bit mutation probability $pm_{\mathrm{bit}} = 0.05$, reposition mutation probability $pm_{\mathrm{reposition}} = 0.50$, and elitist rate $G = 19/140 = 0.14$. The HGA in [18] was originally designed for the MRCPSP with the makespan criterion. The encoding scheme of the HGA is based on domain-based heuristics. In the experiment, we modify the HGA for the MRCPSPDCF with the cash flow criterion by incorporating the MDC heuristic defined in (32) to the algorithm. In the SA and TS approaches [14], the neighborhood is defined by shifting the position of an activity on the activity list or changing the execution mode of an activity. The parameters of the SA and TS are set according to [14] as follows. In the SA, 50 trials of transitions are generated to determine the initial temperature T_0. The initial acceptance ratio χ_0 is set to 0.95 and the distance parameter δ is set to 0.5. In the TS, the length of the tabu list is 7. Parameters for the proposed ACS approach have been given in advance and the hybrid heuristic is used in the comparison. A total of 55 random instances with different sizes are generated. The experiments are run on a machine with Intel Core2 Quad CPU Q6600 at the rate of 2.40 GHz and 1.96 GB of RAM. The operation system is MS Windows XP and the compiler is VC++ 6.0. In the experiment, the algorithms stop when they reach the maximum numbers of NPV evaluations given in Table II.

In Table II, the size of each instance, the maximum evaluation number for each algorithm, the results averaged over 50 runs, and the t-test values are tabulated. Compared with the PGA approach, ACS–MRCPSPDCF yields better results in 47 out of 55 instances. The results of two-tailed t-tests reveal that the ACS approach achieves significantly better results in 37 instances. Compared with the HGA approach, ACS–MRCPSPDCF yields better results in 49 out of 55 instances and achieves significantly better results in 35 instances in terms of the results of two-tailed t-tests. It is important to note that in the largest 15 instances (with 60 activities or more), the proposed approach manages to make significant improvements in all cases. For the SA and TS approaches, as they only search in a neighborhood of the current solution, they are trapped in local optima in some runs. Therefore, according to the results in Table II, the averaged performance of SA and TS is poor compared with the proposed ACS approach.

In Fig. 10, we also make a rough comparison of how fast are the algorithms using CPU time. As the curves of the SA and TS algorithms are out of the scales of the plots, only the curves of ACS, PGA, and HGA are shown in the figure. The results are averaged over 50 runs. According to the results in Fig. 10, in most cases, at the very beginning of the search process, ACS is able to find better solutions than PGA and HGA. This is because the ACS algorithm uses domain-based heuristics to build good initial solutions and searches in an aggressive behavior by setting the parameter q_0 in (14) to 0.9. Furthermore, in the late stages of the search process, ACS is still able to improve the quality of the solutions steadily. These results reveal that the proposed ACS approach outperforms the PGA, HGA, SA, and TS algorithms for the MRCPSPDCF.

V. Conclusion

In this paper, an ACS algorithm for the MRCPSPDCF has been proposed. MRCPSPDCF is an important model for project management applications and the problem is NP-hard. Due to its great complexity, only a few algorithms are available for the problem and the performance is still not satisfying. Compared with the existing approaches, the proposed ACO approach is promising in the following aspects. First, ACO has been found to be good at solving graph-based search problems. In the proposed algorithm, by converting the precedence network (AoA network) into a construction graph (MoN graph), the considered MRCPSPDCF is naturally converted into a graph-based search problem. Therefore ACO is suitable for solving the problem. Second, as the mechanism of ACO supports the use of domain-based heuristics, the proposed algorithm is able to make use of different heuristics to enhance the search ability of ants. Eight heuristics are used in the algorithm, including two precedence-relation-based heuristics, two time-based heuristics, two cost-based heuristics, a resource-based heuristic, and a hybrid heuristic. These heuristics consider different aspects of the tradeoffs in MRCPSPDCF to improve the performance of the algorithm.

In the experiment, the performance of different heuristics is compared. It is found that the hybrid heuristic is able to combine the information from different aspects and achieves the best performance. For the parameters, it is found that $\beta = 1$, $q_0 = 0.9$, and $\psi = 20 - 50$ are able to find good solutions in most cases. The ACS–MRCPSPDCF algorithm is compared with two GA approaches, an SA algorithm and a TS algorithm on 55 random instances. Experimental results reveal that the proposed algorithm outperforms the existing approaches for the MRCPSPDCF.

In the future research, we plan to integrate the ACO algorithm with local refinement procedures for large-scale MRCPSPDCF instances. On the other hand, ACO uses pheromones as a kind of historical records. Such characteristic has been found to be helpful for solving the optimization problems with uncertainties [42]. Therefore, it will be also interesting to apply ACO algorithms to solve the MRCPSPDCF model with uncertain duration or cost.

Acknowledgment

The authors would like to thank the editors and the anonymous reviewers for their valuable comments and suggestions to improve the paper's quality. Jun Zhang is the corresponding author, e-mail: junzhang@ieee.org

References

[1] M. E. Pate-Cornell, G. Tagaras, and K. M. Eisenhardt, "Dynamic optimization of cash flow management decisions: A stochastic model," *IEEE Trans. Eng. Manage.*, vol. 37, no. 3, pp. 203–212, Aug. 1990.

[2] M. Uhrig-Homburg, "Cash flow shortage as an endogenous bankruptcy reason," *J. Banking Finance*, vol. 29, no. 6, pp. 1509–1534, 2005.

[3] W. Herroelen, B. D. Reyck, and E. Demeulemeester, "Resource-constrained project scheduling, a survey of recent developments," *Comput. Oper. Res.*, vol. 13, no. 4, pp. 279–302, 1998.

[4] P. Brucker, A. Drexl, R. Mohring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models and methods," *Eur. J. Oper. Res.*, vol. 112, pp. 3–41, 1999.

[5] A. H. Russell, "Cash flows in networks," *Manage. Sci.*, vol. 16, pp. 357–373, 1970.

[6] R. H. Doersch and J. H. Patterson, "Scheduling a project to maximize its present value, zero-one programming approach," *Manage. Sci.*, vol. 23, pp. 882–889, 1977.

[7] L. Özdamar and G. Ulusoy, "A survey on the resource-constrained project scheduling problem," *IIE Trans.*, vol. 27, pp. 574–586, 1995.

[8] W. Herroelen, B. D. Reyck, and E. Demeulemeester, "Project network models with discounted cash flows: A guided tour through recent developments," *Eur. J. Oper. Res.*, vol. 100, pp. 97–121, 1997.

[9] L. V. Tavares, "A review of the contribution of operation research to project management," *Eur. J. Oper. Res.*, vol. 136, pp. 1–18, 2002.

[10] O. Icmeli and S. S. Erenguc, "The resource constrained time cost tradeoff project scheduling problem with discounted cash flows," *J. Oper. Manage.*, vol. 14, pp. 255–275, 1996.

[11] G. Ulusoy, "Four payment models for the multi-mode resource constrained project scheduling problem with discounted cash flows," *Ann. Oper. Res.*, vol. 102, pp. 237–261, 2001.

[12] J. Blazewicz, J. K. Lenstra, and A. H. G. R. Kan, "Scheduling subject to resource constraints," *Discrete Appl. Math.*, vol. 5, pp. 11–24, 1983.

[13] R. Kolisch, *Project Scheduling Under Resource Constraints, Efficient Heuristics for Several Problem Classes*. Heidelberg, Germany: Physica, 1995.

[14] M. Mika, G. Waligóra, and J. Weglarz, "Simulated annealing andtabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models," *Eur. J. Oper. Res.*, vol. 164, no. 3, pp. 639–668, Aug. 2005.

[15] K. K. Yang, F. B. Talbot, and J. H. Patterson, "Scheduling a project to maximize its net present value: An integer programming approach," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 188–198, Jan. 1993.

[16] O. Icmeli and S. S. Erenguc, "A branch and bound procedure for the resource constrained project scheduling problem with discounted cash flows," *Manage. Sci.*, vol. 42, no. 10, pp. 1395–1408, Oct. 1996.

[17] B. D. Reyck and W. Herroelen, "A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations," *Eur. J. Oper. Res.*, vol. 111, no. 1, pp. 152–174, Nov. 1998.

[18] L. Özdamar, "A genetic algorithm approach to a general category project scheduling problem," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 29, no. 1, pp. 44–59, Feb. 1999.

[19] A. A. Najafi and S. T. A. Niaki, "A genetic algorithm for resource investment problem with discounted cash flows," *Appl. Math. Comput.*, vol. 183, no. 2, pp. 1057–1070, 2006.

[20] Y.-Y. Shou, "Ant colony algorithm for scheduling resource constrained projects with discounted cash flows," in *Proc. 2005 IEEE Int. Conf. Mach. Learning Cyber.*, pp. 176–180.

[21] O. Icmeli and S. S. Erenguc, "A tabu search procedure for the resource constrained project scheduling problem with discounted cash flows," *Comput. Oper. Res.*, vol. 21, no. 8, pp. 841–853, 1994.

[22] R. Etgar, A. Shtub, and L. J. LeBlanc, "Scheduling projects to maximize net present value—The case of time-dependent, contingent cash flows," *Eur. J. Oper. Res.*, vol. 96, no. 1, pp. 90–96, 1997.

[23] A. Kimms, "Maximizing the net present value of a project under resource constraints using a Lagrangian relaxation based heuristic with tight upper bounds," *Ann. Oper. Res.*, vol. 102, pp. 221–236, 2001.

[24] S. Hartmann and R. Kolisch, "Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem," *Eur. J. Oper. Res.*, vol. 127, pp. 394–407, 2000.

[25] R. Kolisch and S. Hartmann, "Experimental investigation of heuristics for resource-constrained project scheduling: an update," *Eur. J. Oper. Res.*, vol. 174, pp. 23–37, 2006.

[26] L. Özdamar, "On scheduling project activities with variable expenditure rates," *IIE Trans.*, vol. 30, no. 8, pp. 695–704, 1998.

[27] G. Ulusoy and S. Cebelli, "An equitable approach to the payment scheduling problem in project management," *Eur. J. Oper. Res.*, vol. 127, pp. 262–278, 2000.

[28] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proc. Eur. Conf. Artif. Life (ECAL 1991)*, NY: Elsevier, pp. 134–142.

[29] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[30] C. Blum and M. Dorigo, "The hyper-cube framework for ant colony optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 34, no. 2, pp. 1161–1172, Apr. 2004.

[31] T. Stützle and H. Hoos, "MAX-MIN ant system and local search for the traveling salesman problem," in *Proc. 1997 IEEE Int. Conf. Evol. Comput. (ICEC)*, Piscataway, NJ: IEEE Press, pp. 309–314.

[32] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to TSP," *IEEE Trans. Evol. Comput*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[33] L. M. Gambardella, É. D. Taillard, and G. Agazzi, "MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows," in *New Ideas in Optimization*. London, U.K.: McGraw-Hill, 1999, pp. 63–76.

[34] K. M. Sim and W. H. Sun, "Ant colony optimization for routing and load-balancing: survey and new directions," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 33, no. 5, pp. 560–572, Sep. 2003.

[35] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

[36] D. Merkle, M. Middendorf, and H. Schmeck, "Ant colony optimization for resource-constrained project scheduling," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 333–346, Aug. 2002.

[37] R. Kolisch and S. Hartmann, "Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis," in *Handbook on Recent Advances in Project Scheduling*, J. Weglarz, Ed. Amsterdam, The Netherlands: Kluwer, 1999, pp. 197–212.

[38] Z. W. He and Y. Xu, "Multi-mode project payment scheduling problems with bonus–penalty structure," *Eur. J. Oper. Res.*, vol. 189, pp. 1191–1207, 2008.

[39] T. Stützle and M. Dorigo, "A short convergence proof for a class of ant colony optimization algorithms," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 358–365, Aug. 2002.

[40] G. Ulusoy and L. Özdamar, "Heuristic performance and network/resource characteristics in resource constrained project scheduling," *J. Oper. Res. Soc.*, vol. 40, pp. 1145–1152, 1989.

[41] A. Drexl and J. Grünewald, "Nonpreemptive multi-mode resource constrained project scheduling," *IIE Trans.*, vol. 25, pp. 74–81, 1993.

[42] P. Balaprakash, "Ant colony optimization under uncertainty," Univ. Libre De Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2005-028, 2005.

Wei-Neng Chen (S'07) received the Bachelor's degree in network engineering in 2006 from Sun Yat-sen University, Guangzhou, China, where he is currently working toward the Ph.D. degree in the Department of Computer Science.

His current research interests include ant colony optimization, genetic algorithms, and other computational intelligence techniques, and also the applications of project management and financial management.

Jun Zhang (M'02–SM'08) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Kowloon, Hong Kong, in 2002.

From 2003 to 2004, he was a Brain Korean 21 Postdoctoral Fellow in the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. Since 2004, he has been with the SUN Yat-sen University, Guangzhou, China. He is currently a Professor in the Department of Computer Science. He has authored four research book chapters and over 60 technical papers in his research areas. His current research interests include computational intelligence, data mining, wireless sensor netowork, operations research, and power electronic circuits.

Prof. Zhang is currently the Chair of the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapter.

Henry Shu-Hung Chung (M'95–SM'03) received the B.Eng. degree in electrical engineering in 1991 and the Ph.D. degree in 1994 from The Hong Kong Polytechnic University, Kowloon, Hong Kong.

Since 1995, he has been with the City University of Hong Kong (CityU), Kowloon. He is currently a Professor in the Department of Electronic Engineering and the Chief Technical Officer of e.Energy Technology Limited—an associate company of CityU. He has authored four research book chapters and over 250 technical papers including 110 refereed journal papers in his research areas, and holds ten patents. His current research interests include time- and frequency-domain analysis of power electronic circuits, switched-capacitor-based converters, random-switching techniques, control methods, digital audio amplifiers, soft-switching converters, and electronic ballast design.

Prof. Chung was the recipient of the Grand Applied Research Excellence Award in 2001 from the CityU. He was the IEEE Student Branch Counselor and was the Track Chair of the Technical Committee on Power Electronics Circuits And Power Systems of the IEEE Circuits and Systems Society from 1997 to 1998. From 1999 to 2003, he was an Associate Editor and a Guest Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, PART I: FUNDAMENTAL THEORY AND APPLICATIONS. He is currently an Associate Editor of the IEEE TRANSACTIONS ON POWER ELECTRONICS AND THE IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS, PART I: FUNDAMENTAL THEORY AND APPLICATIONS.

Rui-Zhang Huang received the M.Phil. and Ph.D. degrees from the Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Hong Kong, in 2003 and 2008, respectively.

Since 2007, she has been with the Hong Kong Polytechnic University, Kowloon, Hong Kong. She has authored or coauthored several papers including those for some prestigious journals and conferences. Her current research interests include data mining, text mining, machine learning, information retrieval, artificial intelligence, and knowledge systems. She has conducted many research works in various fields such as mining unseen named entity translations, semisupervised document clustering, hierarchical incremental document clustering, active learning for clustering, and language modeling, etc.

Ou Liu received the Ph.D. degree in information systems from the City University of Hong Kong, Kowloon, Hong Kong, in 2006.

Since 2006, he has been with the Hong Kong Polytechnic University, Kowloon, where he is currently an Assistant Professor in the School of Accounting and Finance. His current research interests include business intelligence, decision support systems, ontology engineering, genetic algorithms, ant colony system, fuzzy logic, and neural networks.