RAIRO-Oper. Res. 48 (2014) 595–614
DOI: 10.1051/ro/2014025

RAIRO Operations Research
www.rairo-ro.org

AN IMPROVED ANT ALGORITHM FOR MULTI-MODE RESOURCE CONSTRAINED PROJECT SCHEDULING PROBLEM *

Peng Wuliang[1], Huang Min[2] and Hao Yongping[3]

Abstract. Many real-world scheduling problems can be modeled as Multi-mode Resource Constrained Project Scheduling Problems (MRCPSP). However, the MRCPSP is a strong NP-hard problem and very difficult to be solved. The purpose of this research is to investigate a more efficient alternative based on ant algorithm to solve MRCPSP. To enhance the generality along with efficiency of the algorithm, the rule pool is designed to manage numerous priority rules for MRCPSP. Each ant is provided with an independent thread and endowed with the learning ability to dynamically select the excellent priority rules. In addition, all the ants in the ant algorithm have the prejudgment ability to avoid infeasible routes based on the branch and bound method. The algorithm is tested on the well-known benchmark instances in PSPLIB. The computational results validate the effectiveness of the proposed algorithm.

Keywords. Operations Research, Mathematical Programming.

Mathematics Subject Classification. 68M20, 90C59.

Received December 22, 2012. Accepted April 22, 2014.

* *This work was supported by National Science Foundation of China under Grant No. 71071100; the National Science Foundation for Distinguished Young Scholars of China under Grant No. 71325002; the Liaoning BaiQianWan Talents Program under No. 2013921033; the Human Resources Development Fundation of Shenyang under No. 2012141203022.*

[1] School of Economic and Management, Shenyang Ligong University, 110159 Shenyang, P.R. China. peng-wuliang@163.com

[2] College of Information Science and Engineering, Northeastern University, 110819 Shenyang, P.R. China

[3] Laboratory of Advanced Manufacture and Equipment of Liaoning, Shenyang Ligong University, 110159 Shenyang, P.R. China

Article published by EDP Sciences

1. Introduction

Resource Constrained Project Scheduling Problem (RCPSP) involves assigning activities to a set of resources with limited capacity in order to meet some predefined objectives, the most common objective of which is minimizing make-span, *i.e.* minimizing the time to complete the entire project. Thereby, technological precedence constraints have to be observed as well as limitations of the resources required to accomplish the activities. For more information on the RCPSP and solution methods, the reader is referred to [9]. Traditionally, each activity in the RCPSP can be executed in a unique way and it is defined as single mode RCPSP (SRCPSP), which is given by its fixed duration and fixed resource requirements during each period. Researchers have extended the activity/duration concept by allowing for alternative execution modes to complete an activity. Each execution mode reflects a unique, feasible alternative, which combines a duration and resource requirement that allows for accomplishing the underlying activity in various ways. MRCPSP represents these types of problems well.

In MRCPSP, each activity might be accomplished in one of several execution modes, each of which represents an alternative combination of resource requirements of the activity and its duration [12]. The algorithms to solve MRCPSP can be classified into exact algorithms and heuristic algorithms. Although there are considerable efforts on exact algorithms for solving MRCPSP [14, 27, 32], just as drawn by Sprecher and Drexl [28], exact algorithms are unable to find optimal solutions for projects with more than 20 activities and three modes per activity when they are highly resource-constrained. Then heuristics have become the alternative, and the last generation of them, *i.e.* intelligent algorithms, which generate near-optimal schedules for larger projects, is of special interest. In this paper, we focus on investigating a more efficient intelligent algorithm to solve MRCPSP based on ant algorithm.

The remainder of the paper is organized as follows. Section 1 describes the problem and formulates the conceptual model and Section 2 presents a literature review. Section 3 describes our solution for MRCPSP and Section 4 reports the computational experiments, along with comparisons with other existing algorithms. Concluding remarks are made in Section 5.

2. Problem description

In MRCPSP, given the estimated work content for each activity, a set of possible execution modes can be specified for the activity execution. Each mode is characterized by a processing time and an amount of a particular resource type for completing the activity. For example, one worker might finish a job in 8 h (mode 1), whereas two workers might finish the same job in 4 h (mode 2). The product of the activity duration and the amount of the resource type needed is called the activity work content. In the previous example, in modes 1 and 2 the activity has the same work content of 10 worker-hours. Resources available for

completing tasks can be classified as either renewable or non-renewable. The non-renewable resources are depleted after a certain amount of consumption, while renewable resources typically have the same amount of availability in every period for an unlimited number of periods.

Formally, we can describe the MRCPSP as follows. A project consists in J activities. The precedence relationships between activities are defined by a directed acyclic graph $G \in (V, E)$, where V is a set of activities numbered from 1 to J and E is the set of pairs of activities for which a finish-start precedence relationship with time lag 0 exists. The fixed integer duration of an activity is denoted by $d_j (1 \leq j \leq J)$, its integer starting time by s_j and its integer finishing time by f_j. No activity may be started before all its predecessors are finished. Graph G is numerically numbered, *i.e.* an activity always has a higher number than all its predecessors. Each activity $j, j = 1, 2, \ldots J$ has to be executed in one of M_j modes. The activities are non-preemptable and a mode chosen for an activity may not be changed (*i.e.* an activity j started in mode $m \in M_j$ must be completed in mode m without preemption). The duration of activity j executed in mode m is d_{jm}. Assuming that there are R renewable and C non-renewable resources, the number of available units of renewable resource $k, k = 1, 2, \ldots R$ is R_k and the number of available units of non-renewable resource $l, l = 1, 2, \ldots C$ is C_l. Each activity j executed in mode m requires for its processing r_{jmk} renewable resource k, and consumes c_{jml} units of non-renewable resource l. The objective of MRCPSP is to find an assignment of modes to activities as well as precedence and resource-feasible starting times for all activities, such that the make-span of the project is minimized. The conceptual model of MRCPSP is formulated as:

$$\text{Min} \qquad f_n \tag{2.1}$$

$$S.T. \qquad s_j - s_i \geq \sum_{m \in M_j} (x_{im} \cdot d_{im}), (i, j) \in E \tag{2.2}$$

$$\sum_{j \in A_t} \sum_{m \in M_j} (x_{jm} \cdot r_{jmk}) \leq R_k, i \leq t \leq f_n, k = 1, 2, \ldots K \tag{2.3}$$

$$\sum_{j \in V} \sum_{m \in M_j} x_{jm} \cdot c_{jml} \leq C_l, l = 1, 2, \ldots C \tag{2.4}$$

$$\sum_{m \in M_i} x_{im} = 1, i \in V. \tag{2.5}$$

The objective function is given as (2.1), where the project duration is minimized by minimizing the finishing time of the end activity n. The precedence constraints are guaranteed in (2.2), where x_{im} is a decision variable, either valued 1 means activity i is executed in mode m or valued 0 otherwise. (2.3) takes into consideration that the total resource demand of each renewable resource type k does not exceed its availability at each time unit t, where A_t denotes the set of activities being

processed at time instant t. (2.4) limits that the total resource demand of non-renewable resource type l does not exceed its availability at each time unit t. (2.5) guarantees that each activity finishes with only one mode.

3. Literature review

3.1. Heuristics for MRCPSP

As a well-known NP-hard problem, MRCPSP is an important and challenging problem that has gained increasing attention for several years. Nevertheless, it is shown that exact methods cannot solve problems with more than 20 activities executed on three modes and highly resource-constrained. Hence, in practice, heuristics to generate near-optimal schedules of MRCPSP for larger projects are of special interest. Drexl and Grünewald suggested a regret-based biased random sampling approach [11]. Slowinski *et al.* described a single-pass approach, a multi-pass approach and a simulated annealing (SA) algorithm [26]. Boctor presented a heuristic for multi-mode problems without non-renewable resources [5]. Kolisch and Drexl presented a local search procedure [18], and Ozdamar proposed a Genetic Algorithm (GA) based on a priority rule encoding [22]. Hartmann presented a genetic algorithm that uses local search to improve the schedules found by the basic genetic algorithm, and compared the performance of the algorithm with those of other existing heuristics [13]. Alcaraz *et al.* developed a genetic algorithm solving MRCPSP, they extended the representation and operator used in SRCPSP, and defined a new fitness function for infeasible individuals [2]. Bouleimen introduced a Simulated Annealing (SA) heuristic, in which two embedded search loops were used to alternate activity and mode neighborhood exploration [6]. More recently, Ranjbar *et al.* presented a hybrid Scatter Search (SS) for the discrete time/resource trade-off problem that is also a sub-problem of the MRCPSP [25]. Lova *et al.* applied justification technology in MRCPSP, defined as multi-mode forward-backward improvement method, and developed a hybrid genetic algorithm [20]. Peteghem and Vanhoucke applied a bi-population genetic algorithm to solve multi-mode resource-constrained project scheduling problem and its preemptive version [23]. They also presented an Artificial Immune System (AIS) to solve the same problem, their experiment results showed that it achieved the best computational quality until now [24].

3.2. Swarm intelligence for MRCPSP

Swarm intelligence is a discipline that deals with natural and artificial systems composed of many individuals that coordinate using decentralized control and self-organization. Some examples of natural swarm intelligent systems are ant colonies, slime molds, bee and wasp swarms. The research on the swarm intelligence for MRCPSP is relatively scarce. Nevertheless, there have been some literatures involving swarm intelligence for SRCPSP up to now. There are two

popular swarm intelligence algorithms that are the Particle Swarm Optimization (PSO) and the ant algorithm. In general, the PSO is a population-based stochastic approach for solving continuous problem. Since project scheduling problems are discrete problems, there are few studies on PSO solving PSP, the representatives of which are [16,31]. Compared with PSO, the ant algorithm is a kind of algorithm for solving discrete problems in nature, especially for combinatorial optimization problems. Abdallah *et al.* presented an Ant system for calculating both deterministic and probabilistic CPM/PERT networks, in which the resource constraints are not taken into account [1]. Tseng and Chen presented a hybrid meta-heuristic algorithm combining ant colony optimization(ACO, *i.e.* ant algorithm), and GA to solve RCPSP [29]. First, the ant algorithm searches the solution space and generates activity lists to provide the initial population for GA. Next, GA is executed and the pheromone set in ACO is updated based on the solution obtained by GA. And then, ACO searches again by using the new pheromone set. Bautista and Pereira designed three kinds of ant colony algorithms with different solution construction models, which are ACO-T (ACO with trail left between activities), ACO-P (ACO with trail left between activity and position) and ACO-MP (ACO with trail left between activity and positions and read in an accumulative fashion) [3]. Daniel *et al.* developed a colony optimization for resource-constrained project scheduling, in which a combination of two pheromone evaluation methods is used by ants to find new solutions [8]. More recently, Wang *et al.* designed an ACO-SS algorithm that combines a local search strategy, an ant algorithm, and a scatter search (SS) in an iterative process [30]. ACO first searches the solution space and generates activity lists to provide the initial population for the SS algorithm. Then, the SS algorithm builds a reference set from the pheromone trails of the ACO and improves these to obtain better solutions. Thereafter, the ACO uses the improved solutions to update the pheromone set. Finally in this iteration, the ACO searches the solution set using the new pheromone trails after the SS has terminated.

4. Proposed ant algorithm for MRCPSP

The general idea of our solution is using ant algorithm to determine the priority values of activities and their execution modes, which are used to generate an optimal or near optimal schedule by Serial Schedule Generation Scheme (SSGS) or Parallel Schedule Generation Scheme (PSGS). The principle of the algorithm is similar to the algorithm that is widely applied in ant system traveling salesperson problem (AS-TSP), introduced by Dorigo *et al.* [10]. Artificial ants used in the ant algorithm are stochastic solution construction procedures that probabilistically build a solution by iteratively adding solution components to partial solutions by taking into account (i) heuristic information about the problem instance being solved, if available; and (ii) pheromone trails which change dynamically at runtime to reflect the agents acquired search experience. In each iteration, an ant finishes a tour and creates one solution. During the tour, the ant uses heuristic

value as well as pheromone information to select an activity in the activity list. The heuristic value is generated by some problem-dependent priority rule and the pheromone information left by historical tours of the ant colony. The best solution found so far and the best solution in current generation are both used to update the pheromone information. However, before that, some portion of pheromone is evaporated according to the formulation as:

$$\tau_{ij} = (1 - \rho)\,\tau_{ij} \tag{4.1}$$

where ρ is the evaporation rate, j is an activity and i is the position of activity j in the activity list of the best solution found so far. The reason for this is that old pheromone should not have too strong an influence on the future. Then, for every activity $j \in J$, some amount of pheromone is added to element τ_{ij} of the pheromone matrix. This is an elitist strategy that leads ants to search near optimal solution. The amount of pheromone added can be $\rho/2T^*$, where T^* is the make-span of the found best schedule, *i.e.*:

$$\tau_{ij} = \tau_{ij} + \rho\,\frac{1}{2T^*}. \tag{4.2}$$

The same is done also for the best solution found in the current iteration, *i.e.*, for every activity $j \in J$, pheromone is added to τ_{ij}. The algorithm runs until the stopping criterion is met, *e.g.*, a certain number of iterations have been done or the average quality of the solutions found by the ants of a generation has not changed for many iterations.

The initial problem solved by the ant algorithm is Traveling Salesperson Problem (AS-TSP). As mentioned before, although there are a few ant algorithms for RCPSP, all of them are presented to solve SRCPSP, and the effort of ant solving MRCPSP hasn't yet been reported. In this paper, we present an ant algorithm to solve MRCPSP. Compared with existing ones, there are some new features in the solution presented in this paper:

(1) In existing ant algorithms for RCPSP, the solution construction models specify activity lists corresponding to the feasible solutions as tours of ants. Considering the multiple executing mode, we present a new solution construction model, in which the element of pheromone matrix is an activity-mode option.
(2) A general advantage of the ant algorithm is that it is convenient to apply priority rules. However, only few rules can be utilized in current ant systems for solving RCPSP. It is known that a priority rule only adapt to solving special types of project instances, and it is difficult to choose appropriate one from a large number of priority rules in practical cases. In this paper, we present a new approach, named rule pool, to manage and utilize a large number of priority rules.
(3) Each ant is provided with an independent thread and endowed the learning ability. The performances of priority rules are recorded in the rule pool, and they are dynamically updated to make the more efficient priority rules be chosen by more ants.

(4) Ants are endowed the pre-judgment ability to avoid the non-meaningful paths that may generate infeasible plans.

In the following sub sections, we will present the decoding scheme, encoding scheme and solution construction model and other features of the algorithm in detail.

4.1. Decoding

The backbone of most improvement heuristics for solving the RCPSP, where an initial solution is gradually improved, is a schedule representation scheme, a schedule generation scheme and a solution evaluation procedure. The schedule representation can be regarded as an encoding of a schedule. To decode the representation into a schedule, the schedule generation scheme should be executed. In this section, we consider the schedule generation scheme of the presented ant algorithm.

There are two main schedule generation schemes, *i.e.* Serial Schedule Generation Scheme (SSGS) and Parallel Schedule Generation Scheme (PSGS) for project scheduling, which have been discussed by Kolisch [17]. They showed that PSGS does not generally perform better than SSGS, and parallel scheme is possible not to obtain an optimal solution. Therefore, in most existing heuristics for SRCPSP, SSGS is the popular method to decode the representation into a schedule. However, different from that in SRCPSP, in MRCPSP, the execution modes should be assigned for every non-dummy activity. Therefore, the traditional SSGS formulated by Kolisch can not serve to MRCPSP directly. In this paper, the traditional SSGS for SRCPSP is extended for heuristics of MRCPSP.

Assuming that the priority values of activities and their modes have been determined, in the decoding based on SSGS, one activity in a project from the decision set is selected according to activity priority values, and the activity with the highest priority value is chosen and scheduled firstly. In addition, since there are several execution modes in each activity, the execution mode with highest priority value should be also chosen as the scheduled mode for the activity. We select an activity according to the feasible finish time and resource constraints through SSGS and the selected activity is moved from the decision set to the scheduled set with a selected mode. The decision set and the scheduled set keep dynamical updating until the algorithm terminates when all activities are put in the scheduled set at stage J. The decoding procedure to generate schedules through the extended serial schedule generation scheme is formulated as follows:

(1) $h := 1, PS_h := 1$
(2) While $|PS_h| < JDo$
(3) Begin
(4) $\quad E_h = j | j \notin PS_h, P_j \subseteq PS_h$
(5) $\quad j^* \leftarrow j | v_j = \mathrm{Max}_{i \in E_h}\{v_i\}$
(6) $\quad m^* \leftarrow m | v_{jm} = \mathrm{Max}_{k \in M_j}\{v_{jk}\}$
(7) $\quad d_{j^*} := d_{j^* m^*}$

1	2	3	4	5	Positions: Schedule order
1	4	3	2	5	Values :Activity ID

(a) Activity List for SRCPSP

1	2	3	4	5	Positions: Schedule order
1:1	4:2	3:2	2:3	5:1	Values : Activity and mode IDs

(b) Activity List for MRCPSP

FIGURE 1. Examples of encoding based on activity list for MRCPSP.

(8) $ES_{j^*} = \text{Max}\{f_i | i \in P_{j^*}\}$

(9) $s_{j^*} := \text{Min}\{t | ES_{j^*} \leq t, r_{j^*m^*k} \leq vr_{kt}, vr_{kt} = R_k - \sum_{j \in A_t} r_{jk}, \tau = t, t+1, t+2, \ldots, t+d_{j^*}-1, k = 1, 2, \ldots K\}$

(10) $f_{j^*} := s_{j^*} + d_{j^*}$

(11) $PS_{h+1} := PS_h \cup j^*$

(12) $h := h + 1$

(13) *End*

In the algorithm, a schedule procedure is divided into J stages, and only one activity is scheduled at each stage, denoted by integer h, which is also the loop variable. At the initial stage, activity 1 is scheduled, as Step (1), where PS_h is the partial scheduled plan at stage h. Step (2) starts a loop, and it is terminated until all of the activities are scheduled into PS_h at stage J. Step (4) generates the feasible set of activities E_h, in which all the predecessors of j, denoted by P_j, must be scheduled in PS_h to meet the precedence relationships at stage h, and should be updated in each iteration. (5) selects an activity j^* with the highest priority value in E_h, where v_i is the priority value of activity i. (6) assigns a mode m^* with the highest priority value among all of the modes of j^*, where v_{jk} is the priority value of mode k of activity j. (7) values the duration $d_{j^*m^*}$, which is the duration of j^* with m^*. In (8), ES_{j^*} is the earliest start time of activity j^* and valued by the maximum finish time of all the predecessors of j_*. (9) calculates the earliest start time s_{j^*} of j^*. vr_{kt} is the residual availability of renewable resource k in time unit t. (10) calculates the finish time of j^*, denoted by f_{j^*}. (11) moves j^* to PS_h. (12) is the increment expression of the loop variable h. Subsequently, stage h is over and the next stage $h+1$ is started until all the activities are scheduled in PS_h.

4.2. Encoding scheme and solution construction model

A popular encoding scheme for RCPSP in the current researches is permutation encoding, *i.e.* activity list, where all the activities are permutated in a precedence-feasible activity list and the activity closer to the head of the list has the higher priority value and will be scheduled earlier, as shown in Figure 1a. Considering the multiple execution modes in MRCPSP, we define the encoding scheme based on activity list as the example in Figure 1b, where the identities(IDs) of activities and their modes are both encoded in each positions of the activity list.

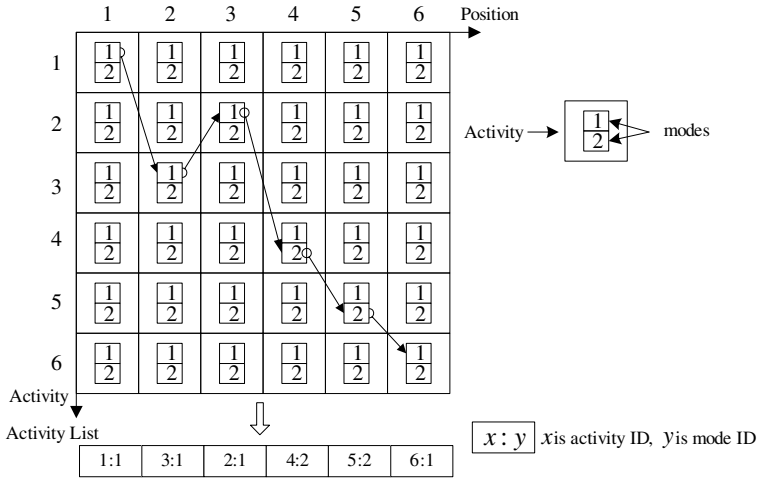

FIGURE 2. Example of solution construction model and encoding individual.

The solution construction model of the ant algorithm can be represented as diagram $D = (V, P, \Gamma)$, where V is activity set, P is position set of ants and Γ is the pheromone distribution. We can regard the solution construction model as a matrix, V as ordinate, P as abscissa, and Γ as the value set in the discrete position located by discrete points in V and P. Correspondingly, $j \in V$ is activity ID, which should be selected during the tours of ants. $i \in P$ denotes a position of an ant's route. τ_{ij} is the amount of pheromone left by ant colony when some of ants selected activity j in step i. As each activity has several modes, the value of τ_{ij} is distributed in these modes, *i.e.*:

$$\tau_{ij} = \sum_{m=1}^{M_j} \tau_{ijm} \tag{4.3}$$

where τ_{ijm} is the pheromone left in execution mode m of activity j in position(step)i.

An example illustrating the solution construction model is shown as Figure 2, where the position is used as abscissa and the activity ID is used as ordinate in a pheromone matrix. An ant moves a step in the matrix, an activity ID and its mode are selected. Therefore, when an ant finishes a complete tour from left to right, it will construct a complete activity list T, in which positions of T suggest priority values, including an activity ID and a mode ID.

4.3. ANT'S TOUR AND INDIVIDUAL BEHAVIOR

Each ant in the presented ant algorithm is an autonomous individual owning an independent thread. The behaviors of an ant follow the steps shown in Figure 3. Firstly, it selects a priority rule from the rule pool as the initial behavior habit,

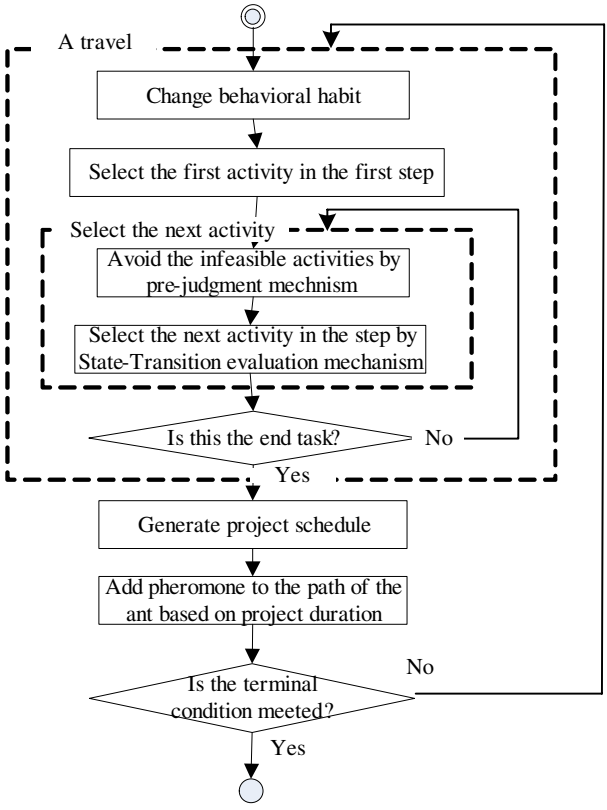

FIGURE 3. Example of an ant's tour.

which can be dynamically changed by choosing another priority rule when a new tour is started. And then, the pre-judgment mechanism is used by ants to choose new nodes. After an ant finished a tour, a complete activity list will be achieved. The activity list is used by the schedule generation scheme described in Section 3.1 to generate entire schedule. The make-span of the schedule generated by each ant is the essential basis of pheromone updating. The rule pool, the mechanism of changing behavior habit, pre-judgment mechanism, evaluation mechanism and pheromone updating mechanism will be elaborated in the following subsections.

4.4. Activity and mode priority rules

In MRCPSP, priority rules are used to choose activity-mode options. Commonly, priority rules consist of two components: a numerical measure and a priority function. Numerical measure is related to properties of the activities, the modes of activities, and resources, and priority function is used to rank activity-modes. Since the problem addressed in this paper is a multi-mode problem, both activities and their execution modes must be considered. The numerical measure used

TABLE 1. Normalization of activity and mode priority values.

Normalized value	Minimum (Min)	Maximum (Max)
Activity priority rule measure	$v_j^{'} = \frac{1/v_j}{\sum_{i \in E_h}(1/v_i)}$	$v_j^{'} = \frac{v_j}{\sum_{i \in E_h}(v_i)}$
Mode priority rule measure	$v_{jk}^{'} = \frac{1/v_{jk}}{\sum_{i \in E_h}(1/v_{ik})}$	$v_{jk}^{'} = \frac{v_{jk}}{\sum_{i \in E_h}(v_{ik})}$

in the activity-mode priority rule should combine the numerical measures of an activity priority rule and that of a mode priority rule. The used activity priority rules formulated and reviewed by Hartmann and Kolish [15] including LFT (Latest Finish Time), LST (Latest Start Time) and MTS (Most Total Successors), have been demonstrated to be effective in single mode problems. The existing mode priority rules, introduced by Boctor [4], includes SFM (Shortest Feasible Mode), LTRU (Least Total Resource Usage) and LCRU (Least Critical Resource Usage). SFM is adopted directly, while LTRU and LCRU are modified to take into account the duration of each activity-mode alternative.

We refer to the approach introduced by Buddhakulsomsiri and Kim [7] to compute the priority values of activity-mode options. The activity-mode priority rule numerical measure is a sum of normalized activity priority rule measures and normalized numerical values from a mode priority rule. Normalization is necessary here to eliminate the differences caused by different units and measures found in the activity and mode priority rules. Let's remind that E_h denotes the set of eligible activities that can be scheduled at some stage h of schedule generation scheme (SGS), v_j denotes the activity priority value and v_{jk} denotes the mode priority value, where $j \in E_h$. Here let $v_j^{'}$ denote the normalized activity priority value and $v_{jk}^{'}$ denote the normalized mode priority value. The normalization is shown in Table 1 and depend on the priority function associated with the activity or mode priority rule. With this normalization, more attractive activities (or modes) with respect to the priority function will have higher normalized priority values.

Assuming that there are three activity priority rules: LFT, LST and MTS, and three mode priority rules: SFM, LTRU and LCRU, 9 different activity-mode priority rule numerical measures will be created, as listed in Table 2. The activity-mode priority rule value is calculated from an activity-mode priority rule combination as follows [7]:

Step 1. For each activity $j \in E_h$, determine the mode k with the highest normalized mode priority value with respect to the selected mode priority rule.
Step 2. Normalize the mode priority values across activities with respect to the mode values selected in Step 1. Call this value m_{jk}^{*}.
Step 3. Calculate the normalized activity priority value with respect to the selected activity priority rule (Tab. 1).
Step 4. Combine the activity priority value and mode priority value into a single numerical measure η_{jk} as follows:

$$\eta_{jk} = a_j^{'} + m_{jk}^{*}. \tag{4.4}$$

TABLE 2. Activity-mode priority rule combinations.

Activity priority rule-mode priority rule		
LFT-LTRU	LST-LTRU	MTS-LTRU
LFT-SFM	LST-SFM	MTS-SFM
LFT-LCRU	LST-LCRU	MTS-LCRU

4.5. Rule pool and the mechanism of changing behavior habit

Rule pool maintains numerous priority rules, which have the identical selection probability to be selected in the initial status but will be dynamically updated according to their performances. Accordingly, the priority rules with higher selection probabilities will be selected by more ants.

ϕ is defined as the rule pool, in which $\xi \in \phi$ is a priority rule numbered $1, 2, \ldots |\phi|$. The priority rule ξ in the rule pool can be expressed by a triple, $\xi = (f, \varphi, u)$. f is the priority calculation function and different from each other among priority rules. φ is the selection probability of a priority rule. A priority rule is randomly selected with the selection probability φ from the rule pool when a tour is finished and the next tour begins. Obviously, if there are $|\varphi|$ rules in the pool, then $\sum_{\xi=1}^{|\phi|} \varphi_\xi = 1$, φ_ξ is the selection probability of rule ξ. Let u_ξ denote the utility coefficient of rule ξ, and it is valued the average of make-spans of schedules generated by this priority rule ξ. The utility coefficients of all the rules are initially valued as sum of durations of all the activities when the first mode is selected and resource constraints are relaxed. When an ant finishes a complete tour, it feedbacks to update the utility coefficient as follows:

$$u_\xi = \frac{u_\xi \cdot w_\xi + L_\xi}{w_\xi + 1} \tag{4.5}$$

where L_ξ is the make-span of the schedule generated by the activity list created by an new ant's tour with priority rule ξ, and w_ξ is the total times of rule ξ being selected so far.

Simultaneously, the selection probabilities of the priority rules in the rule pool are also required to be updated regularly according to utility coefficients of the priority rules. The updating mechanism is formulated as:

$$\varphi_\xi = \frac{u_{\max} - u_\xi + 1/Q}{\sum_{j=1}^{|\phi|}(u_{\max} - u_j + 1/Q)} \tag{4.6}$$

where $u_{\max}$ is the maximum value of the utility coefficient of all rules in the rule pool. Since the objective function is minimizing project duration, the priority rule with lower utility coefficient value has higher probability to be selected through a conversion by $u_{\max} - u_\xi$. In addition, Q is a constant, and $Q > 1$. $1/Q$ is used to guarantee that the rules that have poor performance can still be selected with a lower probability.

The rule pool is classified into two types: one is static rule pool, where all the priority rules have the same and invariant selection probability; the other is dynamic rule pool, which is updated regularly according to (4.6). The interval Δt_r of regularly updating the selection probability in dynamic pool rule can be determined by the number of schedules generated by the algorithm.

4.6. The pre-judgment mechanism

When moving in pheromone matrix, ants might encounter many infeasible routes due to the precedence relationship constraints and/or resource limitations. If these infeasible paths can't be voided in time, the ants will do many meaningless efforts and generate lots of infeasible plans. Therefore, the algorithm efficiency will be remarkably improved if ants avoid those infeasible routes by a pre-judgment mechanism. We designed a pre-judgment mechanism where the branch-and-bound method is used to avoid the infeasible paths by setting the upper bound of resources.

As defined before, G is used to denote the activity set. Furthermore, we let G denote a complete activity list as well, G_κ denote an incomplete activity list generated in the process of an ant's tour at location κ, and $\overline{G_\kappa}$ is composed of activities which belong to G and not to G_κ. The ants can only select new path from feasible activity set E_κ at location κ due to the precedence relationship restraints. The sufficient and necessary condition of $j \in E_\kappa$ is $j \notin G_\kappa$, and $P_j \subset G_\kappa$, where P_j is the predecessors of activity j. Therefore, when an ant moves, it should not select the activity $j \notin E_\kappa$. During step κ of an ant's tour, it owns an incomplete activity list G_κ that can be input into the schedule generation scheme to generate an incomplete project plan.

Definition 4.1. In the step κ, when an ant selects an activity j from the feasible activity set E_κ and put it into G_κ, a new incomplete activity list $G_{\kappa+1}$ is generated. It is assumed that all the activities of $\overline{G_{\kappa+1}}$ select the modes that have the minimum consumption of nonrenewable resources κ. $G_{\kappa+1}$ can be used to generate an incomplete schedule $Y_{\kappa+1}$. Based on $Y_{\kappa+1}$, relaxing the renewable resource constraints of the activities in $\overline{G_{\kappa+1}}$, using the traditional critical path method (CPM) to schedule the activities in $\overline{G_{\kappa+1}}$ on the base of $Y_{\kappa+1}$, a temporary complete schedule Y' can be achieved. The consumption of the nonrenewable resource k in Y' is defined as the *minimum possible resource consumption* of k in the stage $\kappa + 1$, or the *minimum possible resource consumption* of k to G_k for j.

By calculating the minimum possible resource consumption, we design an advanced pre-judgment mechanism referring to the branch and bound method. For a non-renewable resource k, if the minimum possible resource consumption of k to G_k for j exceeds the general amount C^k, then the branch from activity j will be cut. Consequently, the pre-judgment can be classified into two types: one is the basic pre-judgment mechanism, which cut the activities that do not belong to feasible activity set E_κ; the other is advanced pre-judgment, where besides the function of basic pre-judgment mechanism, minimum possible resource

consumption of k to G_k for j must not exceeds C_k as well. In some cases, an ant may find that all the paths are cut by advanced pre-judgment mechanism. In this case, the ant should turn back to the previous position and reselect a new activity.

4.7. State-transition evaluation mechanism

After ants cut infeasible routes according to the pre-judgment mechanism described in the previous subsection, ants evaluate the left routes according to the state-transition evaluation mechanism. The state-transition evaluation is mainly based on two elements: one is ants' individual behavioral habit determined by the priority rule bounded to the ant; the other is the pheromone remained in the pheromone matrix, *i.e.*, pheromone evaluation.

There are two kinds of state-transition evaluation mechanisms: the direct evaluation method and the summation evaluation method. The former evaluates the new location directly, and the later considers both the ant's historical selections of the tour and new positions. The objective of the solution construction model presented in this paper aims to search an activity list which generates a near optimal schedule. It is of no interest to put one special activity into the activity list, and the meaningful thing here is the permutation of all these activities. Therefore, in this paper, the summation evaluation method is employed as the following formulation [21]:

$$p_{ijm} = \frac{\left(\sum_{k=1}^{i} \sum_{m' \in M_k} \left(x_{km'} \cdot \tau_{kjm'}\right)\right)^{\alpha} \cdot \eta_{jm}^{\beta}}{\sum_{h \in E_i} \left[\left(\sum_{k=1}^{i} \sum_{m' \in M_k} \left(x_{km'} \cdot \tau_{kjm'}\right)\right)^{\alpha} \cdot \eta_{jm}^{\beta}\right]} \tag{4.7}$$

where $\sum_{k=1}^{i} \sum_{m' \in M_k} (x_{km'} \cdot \tau_{kjm'})$ is the amount of the pheromone distributed in its trial when it moves to position i. E_i is the set of feasible routes (*i.e.* feasible activity set). η_{jm} is the priority coefficient activity j with mode m, which is calculated based on rules bounded by the ant according to (4.4). Parameter α is the pheromone influence factor determining the influence degree of pheromone to the state-transition evaluation, and parameter β is the influence factor of heuristic information determining the influence degree of priority rules to the state-transition evaluation respectively.

4.8. Pheromone updating mechanism

We design a new pheromone updating mechanism, which has two main features: one is constant-conservation, the other is periodic volatilizing. The so called constant-conservation is that the total quantity of pheromone keeps constant, and the ratio between pheromone quantity remained in the distribution matrix and that volatilized in air is also constant after each time of volatilization. The so called periodic volatilizing is that the pheromone volatilizes periodically at a certain time interval. The time interval of pheromone volatilization can be defined by the number of generated schedules (the concept is the same as the updating interval Δt_r described in Sect. 3.5).

Let U denote the amount of the pheromone, U_r denote the remained pheromone in the pheromone matrix, and U_ρ denote the volatilized pheromone in air. Although U_r and U_ρ is dynamically updated, due to constant-conservation, U is always unchanged, that is, $U = U_r + U_\rho$. After each volatilizing has just completed, $U_r = (1-\rho)\cdot U$ and $U_\rho = \rho \cdot U$ are guaranteed by the algorithm, where ρ is the general volatilization factor used to keep the dynamic balance of the ratio of the U_r and U_ρ. The pheromone of each location in the pheromone matrix volatilizes with the same ratio $\rho^{'}$, as shown in the following equation:

$$\sum_{i=1}^{J}\sum_{j=1}^{J}\sum_{m=1}^{M_j}(\tau_{ijm}\cdot\rho^{'}) = (1-\rho)\cdot U. \tag{4.8}$$

It is seen that $\rho^{'}$ is not a constant during the algorithm execution process and requires to be determined by the pheromone value remained in pheromone matrix and ρ. It is easily drawn from (4.8) that:

$$\rho^{'} = \frac{(1-\rho)\cdot U}{\sum_{i=1}^{J}\sum_{j=1}^{J}\sum_{m=1}^{M_j}\tau_{ijm}}. \tag{4.9}$$

Consequently, the pheromone volatilizing is formulated as:

$$\tau_{ijm} \leftarrow \rho^{'}\cdot\tau_{ijm} = \frac{(1-\rho)\cdot U}{\sum_{i=1}^{J}\sum_{j=1}^{J}\sum_{m=1}^{M_j}\tau_{ijm}}\cdot\tau_{ijm}, 1 \le m \le M_j. \tag{4.10}$$

When finishing a complete tour, the ant individual leaves pheromone in the pheromone matrix according to the pheromone increase mechanism immediately. Then it continues its next tour without waiting for other ants. The pheromone increase mechanism is formulated as:

$$\tau_{ijm} \leftarrow \tau_{ijm} + \Delta\tau_{ij}, 1 \le m \le M_j \tag{4.11}$$

where $\Delta\tau_{ij}$ is the pheromone increment for τ_{ijm}, it indicates that the value of the pheromone only increase to the selected model m of activity j on position i. $\Delta\tau_{ij}$ is calculated according to the following formulation:

$$\Delta\tau_{ij} = \begin{cases} \dfrac{F_w - F^k}{F_w - F_e}\cdot\dfrac{\rho\cdot U}{\Delta t_\rho}, F_w \ne F_e \\ \dfrac{(1-\rho)\cdot U}{\Delta t_\rho}, F_w = F_e \end{cases} \tag{4.12}$$

where F_w is the currently found best solution, F_e is currently found worst solution, and Δt_ρ is the interval of the pheromone volatilizing. For example, $\Delta t_p = 10$ means that it volatilizes once just when 10 ants finished their tours. Apparently, this method is adaptive for the concurrent computation presented in this paper. The increase and volatilization of the pheromone are no more determined by iterations of algorithm, but by the amount of tours of all the ants. In MRCPSP, the amount of the tours of ants is equal to the number of the generated schedules.

5. Computational experiments

5.1. Experimental environment and project instances

The ant algorithm presented in this paper as well as the extended serial schedule generation scheme are programmed in Java, and experiment has been conducted under the Windows XP operating system on a Pentium IBM-compatible PC with 1 CPU, Intel 2.0 GHz, 512 MB RAM, 80 GB Hard Disk. We use the shared sets in PSPLIB [19] as test instances. They were generated using PROGEN under a full factorial experimental design with the following three independent problem parameters: network complexity, resource factor and resource strength [19]. In PSPLIB, there are seven sets of test instances for MRCPSP containing instances with 10, 12, 14, 18, 20, and 30 non-dummy activities. Each of the non-dummy activities may be performed in one out of three modes. The duration of a mode varies between 1 and 10 periods. For each project size, 640 instances were generated. Those instances with no feasible solution have not been considered. Hence, there are 536 instances with $J = 10$, 547 instances with $J = 12$, 551 instances with $J = 14$, 550 instances with $j = 16$, 552 instances with $J = 18$, and 554 instances with $J = 20$. The set with 20 non-dummy activities currently is the hardest standard set of multi-mode instances for which all optimal solutions are known [28]. For the set with 30 activities, not all optimal solutions are known until now, and for some instances it is currently not known if a feasible solution exists.

5.2. Parameters setting

In the algorithms presented in this paper, the main parameters include pheromone influence factor α, influence factor of heuristic information β, population size POP, evaporation rate ρ, time interval of pheromone evaporation Δt_ρ, time interval of rule pool updating Δt_r. The absolute value of either α or β have no meaning, and the meaningful thing is the ratio between them. Let γ denote the ratio of α to β, that is $\gamma = \beta/\alpha$, and thus γ is an other parameter of the algorithm. We evaluate how all the parameters affect the performance of the algorithm by computational experiments. From computational experiments, it is verified when Δt_p and Δt_r is more than 20, and $\gamma = 2$, $\rho = 0.4$, the algorithm achieves better performance. Although POP has little effect on the number of generated schedules to find the optimal solution, different values of POP cause different CPU times of the algorithm seriously. With the increasing of POP, the average CPU time decreases significantly when $POP < 40$. When $POP > 40$, the CPU time will reach a stable level, and the performance of the algorithm is no longer improved. The reason for it is that in the presented algorithm, each ant owns one thread, and more threads will make full use of the processing ability of CPU. However, all the threads share the pheromone matrix and priority rule pool, such that the synchronization mechanism should be employed to ensure the thread-safe. Therefore, too many ant individuals will cause thread waiting when volatilizing pheromone as well as updating rule pool. For the given hardware configuration of the test computer, the algorithm performance will not be improved

TABLE 3. Performances of algorithms with different configurations under 5000 schedules generated.

Instance set	Criteria	Algorithm I	Algorithm II	Algorithm III
J12	Avg dev	0.79%	0.25%	0.09%
	Max dev	–	18.4%	11.6%
	Opt rate	57.2%	87.4%	98.4%
	Inf rate	48.2%	0%	0%
J16	Avg dev	0.98%	0.91%	0.21%
	Max dev	–	27.2%	21.6%
	Opt rate	55.0%	76.4%	89.4%
	Infeasible solution rate	47.1%	0%	0%
J20	Avg dev	1.18%	1.09%	0.54%
	Max dev	–	37.2%	29.6%
	Opt rate	46.0%	68.4%	80.4%
	Inf rate	49.5%	0%	0%

continually with too many threads. Appropriate population size of the ant algorithm presented in this paper is related to computer hardware configuration. In our computer platform, POP is valued 40. For the parameter Q used in (11), it is randomly valued 2 since we found that the parameter has minor influence on the algorithm through experiments.

5.3. The comparisons of experimental results with different configurations

In order to investigate the effectiveness of the extended serial schedule generation scheme, rule pool and prejudgment mechanism, we set three different configurations for the algorithm: algorithm I adopts the basic prejudgment mechanism and the static priority rule pool; algorithm II adopts the advanced prejudgment mechanism and the static rule pool; algorithm III adopts the advanced prejudgment mechanism and the dynamic rule pool. Four criteria are taken into account: average deviation, maximal deviation, optimal solution rate and infeasible solution rate, which are defined as follows:

- **Average deviation(Avg dev)**: Average deviations from the best known solution for instance set.
- **Maximal deviation(Max dev)**: The maximal deviation from the best known solution for instance set.
- **Optimal solution rate(Opt rate)**: The percentage of instances for which an optimal solution was found.
- **Infeasible solution rate(Inf rate)**: The percentage of schedules which are infeasible in all the schedules created during the process of computation.

We select J12, J14 and J20 as test sets to compare the performances of the three algorithms with an upper limit of 5000 generated schedules. Some instances without feasible solution for MRCPSP could be found and they have been excluded in the computational experiment. The computational results are listed in Table 3.

TABLE 4. Comparative computational results for J10.

Authors	Algorithm	Avg dev	Opt rate	Schedules	CPU time
Kolisch *et al.* [18]	Local search	0.50%	91.8%	6000	–
Ozdamar [22]	GA	0.86%	88.1%	6000	–
Hartmann [13]	GA	0.10%	98.01%	6000	–
Alcaraz *et al.* [2]	GA	0.19%	96.5%	6000	0.19[a]
Bouleimen *et al.* [3]	SA	0.21%	96.3%	6000	19.3[b]
Lova *et al.* [20]	GA	0.04%	99.7%	6000	0.1[c]
Lova *et al.* [20]	GA	0.06%	98.51%	5000	0.08[c]
Lova *et al.* [20]	GA	0.08%	98.13%	4000	0.7[c]
Ranjbar *et al.* [25]	SS	0.18%	–	–	10[c]
Peteghem *et al.* [24]	AIS	0.02%	99.4%	5000	–
This work	Ant Algorithm	0.03%	99.7%	6000	0.07
This work	Ant Algorithm	0.03%	99.6%	5000	0.05
This work	Ant Algorithm	0.05%	99.3%	4000	0.05

[a] Pentium with 1.13 GHz and 256 MB RAM.
[b] Pentium with 100 MHz and 32 M RAM.
[c] Pentium with 3 GHz and 1 GB RAM.

Since the basic prejudgment mechanism is used in algorithm I, a large number of generated schedules may not satisfy the non-renewable resource constraints, and the infeasible solution rate is high in algorithm I. Once the advanced prejudgment mechanism is employed in algorithm II and algorithm III, the infeasible solution can not be created, and the computation performence is accordingly enhanced. Furthermore, the rule pool used in algorithm III can be dynamically updated according to rules' performance, and the excellent rules can be automatically found and bound with a higher probability. As a result, the quality and computation efficiency of algorithm III is improved remarkably.

5.4. The comparisons of experimental results with other methods

In current research of MRCPSP, the CPU time is one of performance criteria. However, the algorithms designed by different researchers can not be executed in the same hardware and software configurations, and thus different algorithms can not be compared only with CPU time. It is common practice to compare heuristics by comparing the quality of the best schedules obtained when the maximum number of schedules that can be generated is limited. This practice allows the comparison of results obtained in different configurations. We compare the algorithm III presented in this paper with the reported algorithms, and the results for J10 are listed as Table 4. It is shown that the algorithm presented in this paper outperforms the existing ones in most aspects.

In the reported literatures, more instances set are commonly tested under 5000 schedules generated, that is, the maximum number of schedules that can be generated is limited to a maximum of 5000. To completely compare the performance of this algorithm with the existing ones, we test the Algorithm III using J12, J14, J16, J18, J20 and J30 as instances set under 5000 schedules generated, and compare

TABLE 5. Average deviation under 5000 schedules generated.

Authors	J12	J14	J16	J18	J20	J30
Alcaraz *et al.* (2003) [2]	0.73	1.00	1.12	1.43	1.91	–
Ranjbar *et al.* (2008) [25]	0.65	0.89	0.95	1.21	1.64	–
Jarboui *et al.* (2008) [16]	0.09	0.36	0.44	0.89	1.10	2.35
Peteghem *et al.* (2009) [24]	0.07	0.20	0.39	0.52	0.70	1.55
This work	0.09	0.17	0.34	0.41	0.53	1.38

the computational result with other heuristics in Table 5. It is shown that the performance of ACO algorithm presented in this paper has excellent performance.

6. Conclusion

According to the characteristics of MRCPSP, we presented an improved ACO. A solution construction model of the algorithm can intuitionally describe the pheromone matrix, where each cell is corresponding to an activity-mode option. Since each ant in the new ant algorithm has been assigned an independent thread, it has more autonomous ability and the computing capability of computers can be fully utilized. The rule pool is employed to manage a large number of priority rules as different habits of ants. Moreover, ants in the algorithm are endowed with some prejudgment ability to avoid infeasible routes. The prejudgment mechanism make the algorithm avoid infeasible schedules and thus the computation efficiency of algorithm is remarkably enhanced. The results of computational experiments verified that the features, *i.e.* prejudgment mechanism, rule pool, *etc.* not only improve the computation efficiency of the algorithm, but also enhance the computation quality and generality as well. We believe that the introduction of this algorithm and all its features will provide some references to solving other project scheduling problem as well as other combination optimization problems by ant algorithms.

References

[1] H. Abdallah, H.M. Emara, H.T. Dorrah and A. Bahgat, Using Ant Colony Optimization algorithm for solving project management problems. *Exp. Syst. Appl.* **36** (1999) 10004–10015.

[2] J. Alcaraz, C. Maroto and R. Ruiz, Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *J. Oper. Res. Soc.* **54** (2003) 614–626.

[3] J. Bautista and J. Pereira, Ant colonies for the RCPS problem. *Lect. Notes in Comput. Sci.* **2504** (2002) 257–268.

[4] F.F. Boctor, Heuristics for scheduling projects with resource restrictions and several resource-duration modes. *Int. J. Prod. Res.* **31** (1993) 2547–2558.

[5] F.F. Boctor, A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. *Eur. J. Oper. Res.* **90** (1996) 349–361.

[6] K. Bouleimen and H. Lecocq, A new efficient simulated annealing algorithm for the resource constrained project scheduling problem. *Eur. J. Oper. Res.* **149** (2003) 268–281.

[7] J. Buddhakulsomsiri and D.S. Kim, Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. *Eur. J. Oper. Res.* **178** (2007) 374-390.

[8] M. Daniel, M. Martin and S. Hartmut, Ant colony optimization for resource-constrained project scheduling. *IEEE Trans. Evol. Comput.* **6** (2002) 333–346.
[9] E. Demeulemeester and W. Herroelen, *Project scheduling: A research handbook*. Kluwer Academic Publishers (2002).
[10] M. Dorigo, V. Maniezzo and A. Colorni, Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern* **26** (1996) 29–41.
[11] A. Drexl and J. Grünewald, Nonpreemptive multi-mode resource-constrained project scheduling. *IIE Trans.* **25** (1993) 74–81.
[12] S.E. Elmaghraby, *Activity networks: project planning and control by network models*. Wiley, New York (1997).
[13] S. Hartmann, Project scheduling with multiple modes: a genetic algorithm. *Annal. Oper. Res.* **102** (2001) 111–135.
[14] S. Hartmann and A. Drexl, Project scheduling with multiple modes: a comparison of exact algorithms. *Networks* **32** (1998) 283–297.
[15] S. Hartmann and R. Kolish, Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **127** (2000) 394–407.
[16] B. Jarboui, N. Damak and P. Siarry, A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Appl. Math. Comput.* **195** (2008) 299–308.
[17] R. Kolisch, Serial and parallel resource-constrained project scheduling methods revisited theory and computation. *Eur. J. Oper. Res.* **90** (1996) 320–333.
[18] R. Kolisch and A. Drexl, Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Trans.* **29** (1997) 987–999.
[19] R. Kolisch and A. Sprecher, PSPLIB-A project scheduling problem library. *Eur. J. Oper. Res.* **96** (1997) 205–216.
[20] A. Lova, P. Tormos, M. Cervantes and F. Barber, An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *Int. J. Prod. Econ.s* **117** (2009) 302–316.
[21] D. Merkle and M. Middendorf, An ant algorithm with a new pheromone evaluation rule for total tardiness problems. *Lect. Notes Comput. Sci.* **1803** (2000) 287–296.
[22] L. Ozdamar, A genetic algorithm approach to a general category project scheduling problem. *IEEE Trans. Syst. Man Cybern.* **29** (1999) 44–59.
[23] V.V. Peteghem and M. Vanhoucke, A genetic algorithm for the preemptive and non-preemtive multi-mode resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **201** (2009) 409–418.
[24] V.V. Peteghem and M. Vanhoucke, An artificial immune system for the multi-mode resource-constrained project scheduling problem. *Evol. Comput. Comb. Optim.* **5482** (2009) 85–96.
[25] M. Ranjbar, B. De Reyck, B. and F. Kianfar, A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *Eur. J. Oper. Res.* **193** (2008) 35–48.
[26] R. Slowinski, B. Soniewicki and J. Weglarz, DSS for multiobjective project scheduling subject to multiple-category resource constraints *Eur. J. Oper. Res.* **79** (1994) 220–229.
[27] A. Sprecher, S. Hartmann and A. Drexl, An exact algorithm for the project scheduling with multiple modes. *OR Spektrum* **19** (1997) 195–203.
[28] A. Sprecher and A. Drexl, Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm *Eur. J. Oper. Res.* **107** (1998) 431–450.
[29] L.Y. Tseng and S.C. Chen, A hybrid meta-heuristic for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.* **175** (2006) 707–721.
[30] C. Wang, *et al.*, An efficient hybrid algorithm for resource-constrained project scheduling. *Inform. Sci.* **180** (2010) 1031–1039.
[31] H. Zhang, H. Li and C.M. Tam, Particle swarm optimization for resource-constrained project scheduling. *Int. J. Project Manag.* **24** (2006) 83–92.
[32] G. Zhu, J. Bard and G. Tu, A Branch-and-Cut Procedure for the Multimode Resource-Constrained Project-Scheduling Problem. *J. Comput.* **18** (2006) 377–390.