Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 195 (2008) 299–308
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems

B. Jarboui [a], N. Damak [a], P. Siarry [b,*], A. Rebai [c]

[a] FSEGS, route de l'aéroport km 4, Sfax 3018, Tunisia
[b] LiSSi, Université de Paris 12, 61 avenue du Général de Gaulle, 94010 Créteil, France
[c] ISAAS, route de l'aéroport km 4, B.P. No. 1013, Sfax 3018, Tunisia

Abstract

The particle swarm optimization (PSO) has been widely used to solve continuous problems. The discrete problems have just begun to be also solved by the discrete PSO. However, the combinatorial problems remain a prohibitive area to the PSO mainly in case of integer values. In this paper, we propose a combinatorial PSO (CPSO) algorithm that we take up challenge to use in order to solve a multi-mode resource-constrained project scheduling problem (MRCPSP). The results that have been obtained using a standard set of instances, after extensive experiments, prove to be very competitive in terms of number of problems solved to optimality. By comparing average deviations and percentages of optima found, our CPSO algorithm outperforms the simulated annealing algorithm and it is close to the PSO algorithm.

Keywords: Combinatorial optimization; Combinatorial particle swarm optimization; Multi-mode resource-constrained project scheduling problem; Local search

1. Introduction

Resource-constrained project scheduling problems (RCPSP) involve assigning jobs or tasks to a resource or set of resources with limited capacity, in order to meet some predefined objective. Many different objectives are possible and they depend on the goals of the decision maker, but the most common of them is to find the minimum makespan, i.e. the minimum time to complete the entire project. Thereby, technological precedence constraints have to be observed as well as limitations of the renewable resources required to accomplish the activities. Once started, an activity may not be interrupted. Consequently, we have a more realistic model which is the resource-constrained project scheduling problem with multiple execution modes.

The multi-mode resource-constrained project scheduling problem (MRCPSP) is a NP-hard problem which has been widely studied in the literature. The activities of a project not only have to be scheduled in order to

* Corresponding author.
E-mail address: siarry@univ-paris12.fr (P. Siarry).

0096-3003/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2007.04.096

minimize the project makespan but also they are subject to precedence constraints and limited availability of the resources. Moreover, each activity can be performed in one out of several modes, which consists in a different combination of resource requirements and duration (Elmaghraby [1]).

Several different techniques have been proposed to solve this problem. However, as shown by Sprecher [2] and Sprecher and Drexl [3], exact methods are unable to find optimal solutions for projects with more than 20 activities and three modes per activity when they are highly resource-constrained. Heuristic methods have become the alternative, and the last generation of them, the metaheuristics, are being successfully applied to this problem. Hence, in practice, heuristic algorithms to generate near-optimal schedules for larger projects are of special interest.

Several heuristic procedures for solving the MRCPSP have been proposed in the literature: Drexl and Grünewald [4] suggested a regret-based biased random sampling approach. Slowinski et al. [5] described a single-pass approach, a multi-pass approach, and a simulated annealing algorithm. Kolisch and Drexl [6] presented a local search procedure.

Özdamar [7] proposed a genetic algorithm based on a priority rule encoding. Bouleimen and Lecocq [8] suggested a simulated annealing heuristic. Sprecher and Drexl [3] developed a branch-and-bound procedure which is, according to the results obtained by Hartmann and Drexl [9], the currently most powerful algorithm for exactly solving the MRCPSP. Sprecher and Drexl [3] suggested using it as a heuristic by imposing a time limit. Finally, Boctor [10,11] presented heuristics for multi-mode problems without non-renewable resources.

PSO simulates the behaviours of bird flocking. Suppose the following scenario: a group of birds are randomly searching for food in an area. There is only one piece of food in the area being searched. All of birds do not know where the food is. But they know how far the food is in each iteration. So what is the best strategy to find the food? The effective one is to follow the bird that is nearest to the food.

PSO developers learned from this scenario and used it to solve the optimization problems. In PSO, each single solution is a ''bird'' in the search space. We call it ''particle''. All the particles have fitness values that are evaluated by the fitness function to be optimized, and have velocities that direct the flying of the particles. The particles fly through the problem space by following the current optimum particles. A new application of PSO for solving combinatorial optimization problems has been developed, such as single machine total weighted tardiness problems by Tasgetiren et al. [12], permutation flowshop scheduling problem by Tasgetiren et al. [13] and Liao et al. [14]. Salman et al. [15] have already solved the task assignment problem by the discrete particle swarm optimization but they have just rounded off the values. Zhang et al. [16] provided an alternative methodology to solve the MRCPSP by utilizing the features of PSO. The results of this PSO algorithm will be used to validate our results.

In this paper, we take up a challenge to apply a new combinatorial particle swarm optimization (CPSO) algorithm to solve a large variety of combinatorial optimization problems, namely the multi-mode resource-constrained project scheduling problem using integer values.

The remainder of the paper is organized as follows. Section 2 describes the MRCPSP problem. Section 3 defines the classical PSO algorithm. Section 4 focuses on the proposed combinatorial PSO (CPSO). In Section 5, we adapt our combinatorial PSO to the MRCPSP problem. In Section 6, we describe how the local search optimization can optimize the sequences associated to particles. Experimental results and comparisons are reported in Section 7. Finally, Section 8 concludes the paper.

2. Problem description

In a multi-mode RCPSP (MRCPSP), given the estimated work content for an activity, a set of allowable execution modes can be specified for the activity execution. Each mode is characterized by a processing time and amount of a particular resource type for completing the activity. For example, one worker might finish a job in 10 h (mode 1), whereas two workers might finish the same activity in 5 h (mode 2). The product of the duration of the activity and the amount of the resource type needed is called the activity *work content*. In our previous example, in modes 1 and 2 the activity had a work content of 10 worker-hours.

Resources available for completing tasks can be classified as either renewable or non-renewable. Recall that non-renewable resources are depleted after a certain amount of consumption (or number of periods), 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 relations between activities are defined by a directed acyclic graph $\boldsymbol{G}$. No activity may be started before all its predecessors are finished. Graph $\boldsymbol{G}$ is numerically numbered, i.e. an activity has always a higher number than all its predecessors. Each activity j, $j = 1 \ldots J$ 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, $j = 1 \ldots J$, started in mode, $m \in 1 \ldots M_j$ must be completed in mode m without preemption). The duration of activity j executed in mode m is d_{jm}. We assume that there are R renewable and non-renewable resources. The number of available units of renewable resource k, $k = 1 \ldots R$ is R_k and the number of available units of non-renewable resource l, $l = 1 \ldots N$, is N_l. Each activity j executed in mode m requires for its processing r_{jmk} units of renewable resource k, $k = 1 \ldots R$, and consumes n_{jml} units of non-renewable resource l, $l = 1 \ldots N$. We assume that all activities and resources are available at the beginning of the process. The objective of the 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 makespan of the project is minimized.

3. PSO algorithm

PSO introduced by Kennedy and Eberhart [17] is one of the most recent and hopeful evolutionary metaheuristics, which is inspired from the swarming behaviour of animals and human social behaviour.

Scientists found that the synchrony of animal's behaviour was through maintaining optimal distances between individual members and their neighbours. Thus, velocity plays the important role of adjusting each other for the optimal distance. Furthermore, scientists simulated the scenario in which birds search for food and observed their social behaviour. They perceived that in order to find food the individual members determined their velocities by two factors, their own best previous experience and the best experience of all other members. This is similar to the human behaviour in making decision where people consider their own best past experience and the best experience of the other people around them.

The general principles for the PSO algorithm are stated as follows.

Similarly to evolutionary computation technique, the PSO maintains a population of particles, where each particle represents a potential solution to an optimization problem. Let K be the size of the swarm. Each particle i can be represented as an object with several characteristics.

Suppose that the search space is n-dimensional, then the ith particle can be represented by a n-dimensional vector, $X_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$, and velocity $V_i = \{v_{i1}, v_{i2}, \ldots, v_{in}\}$, where $i = 1, 2, \ldots, K$.

In PSO, particle i remembers the best position it visited so far, referred to as $P_i = \{p_{i1}, p_{i2}, \ldots, p_{in}\}$, and the best position of the best particle in the swarm, referred to as $G = \{G_1, G_2, \ldots, G_n\}$.

The PSO is similar to evolutionary computation algorithm and, in each generation t, particle i adjusts its velocity v_{ij}^t and position x_{ij}^t through each dimension j by referring to, with random multipliers, the personal best position p_{ij}^{t-1} and the swarm's best position G_j^{t-1} using Eqs. (1) and (2) as follows:

$$v_{ij}^t = v_{ij}^{t-1} + c_1 r_1 (p_{ij}^{t-1} - x_{ij}^{t-1}) + c_2 r_2 (G_j^{t-1} - x_{ij}^{t-1}), \tag{1}$$

and

$$x_{ij}^t = x_{ij}^{t-1} + v_{ij}^t, \tag{2}$$

where c_1 and c_2 are the acceleration constants and r_1 and r_2 are random real numbers drawn from $U(0,1)$. Thus the particle flies through potential solutions toward P_i^t and G^t in a navigated way while still exploring new areas by the stochastic mechanism to escape from local optima. Since there was no actual mechanism for controlling the velocity of a particle, it was necessary to impose a maximum value $V_{\max}$ on it. If the velocity exceeded this threshold, it was set equal to $V_{\max}$, which controls the maximum travel distance in each iteration to avoid this particle flying past good solutions.

The PSO algorithm is terminated with a maximal number of generations or the best particle position of the entire swarm that cannot be improved further after a sufficiently large number of generations.

The aforementioned problem was addressed by incorporating a weight parameter for the previous velocity of the particle. Thus, in the latest versions of the PSO, Eqs. (2) and (3) are changed to the following ones:

$$v_{ij}^t = \chi(\omega v_{ij}^{t-1} + c_1 r_1(p_{ij}^{t-1} - x_{ij}^{t-1}) + c_2 r_2(G_j^{t-1} - x_{ij}^{t-1})), \tag{3}$$

$$x_{ij}^t = x_{ij}^{t-1} + v_{ij}^t, \tag{4}$$

where ω is called inertia weight and is employed to control the impact of the previous history of velocities on the current one. Accordingly, the parameter ω regulates the trade-off between the global and local exploration abilities of the swarm. A large inertia weight facilitates global exploration, while a small one tends to facilitate local exploration. A suitable value for the inertia weight ω usually provides balance between global and local exploration abilities and consequently results in a reduction of the number of iterations required to locate the optimum solution.

χ is a constriction factor, which is used to limit velocity.

The PSO algorithm has shown its robustness and efficacy in solving function value optimization problems in real number spaces, only a few researches have been conducted for extending PSO to combinatorial optimization problems under a binary form.

4. Proposed combinatorial PSO (CPSO)

In this paper, we propose an extension of PSO algorithm to solve the combinatorial optimization problem with integer values.

Combinatorial PSO essentially differs from the original (or continuous) PSO in some characteristics.

4.1. Definition of a particle

Denote by $Y_i^t = \{y_{i1}^t, y_{i2}^t, \ldots, y_{in}^t\}$, the n-dimensional vector associated to the solution $X_i^t = \{x_{i1}^t, x_{i2}^t, \ldots, x_{in}^t\}$ taking a value in $\{-1, 0, 1\}$ according to the state of solution of the ith particle at iteration t.

Y_i^t is a dummy variable used to permit the transition from the combinatorial state to the continuous state and vice versa

$$y_{ij}^t = \begin{cases} 1, & \text{if } x_{ij}^t = G_j^t, \\ -1, & \text{if } x_{ij}^t = p_{ij}^t, \\ -1 \text{ or } 1, & \text{randomly if } (x_{ij}^t = G_j^t = p_{ij}^t), \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

4.2. Velocity

Let $d_1 = -1 - y_{ij}^{t-1}$ be the distance between x_{ij}^{t-1} and the best solution obtained by the ith particle.

Let $d_2 = 1 - y_{ij}^{t-1}$ be the distance between the current solution x_{ij}^{t-1} and the best solution obtained in the swarm.

The update equation for the velocity term used in the CPSO is then:

$$v_{ij}^t = w.v_{ij}^{t-1} + r_1.c_1.d_1 + r_2.c_2.d_2, \tag{6}$$

$$v_{ij}^t = w.v_{ij}^{t-1} + r_1.c_1(-1 - y_{ij}^{t-1}) + r_2.c_2(1 - y_{ij}^{t-1}). \tag{7}$$

With this function the variation of the velocity v_{ij}^t depends on the result of y_{ij}^{t-1}.

If $x_{ij}^{t-1} = G_j^{t-1}$, then $y_{ij}^{t-1} = 1$. Thereafter, d_2 turns to "0", and d_1 receives "-2", which is going to impose to the velocity to move in the negative sense.

If $x_{ij}^{t-1} = p_{ij}^{t-1}$, then $y_{ij}^{t-1} = -1$. Thereafter, d_2 turns to "2", and d_1 receives "0", thus imposing to the velocity to move in the positive sense.

In the case $x_{ij}^{t-1} \neq G_j^{t-1}$ and $x_{ij}^{t-1} \neq p_{ij}^{t-1}$, y_{ij}^{t-1} turns to "0", d_2 is equal to "1" and d_1 is equal to "-1", thereafter the parameters r_1, r_2, c_1 and c_2 will determine the sense of the variation of the velocity.

In the case $x_{ij}^{t-1} = p_{ij}^{t-1}$ and $x_{ij}^{t-1} = G_j^{t-1}$, y_{ij}^{t-1} takes a value in $\{-1, 1\}$, thus imposing to the velocity to move in the inverse sense of the sign of y_{ij}^t.

4.3. Construction of a particle solution

The update of the solution is computed within y_{ij}^t:

$$\lambda_{ij}^t = y_{ij}^{t-1} + v_{ij}^t. \tag{8}$$

The value of y_{ij}^t is adjusted with the following function:

$$y_{ij}^t = \begin{cases} 1, & \text{if } \lambda_{ij}^t > \alpha, \\ -1, & \text{if } \lambda_{ij}^t < -\alpha, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The new solution is then

$$x_{ij}^t = \begin{cases} G_j^{t-1}, & \text{if } y_{ij}^t = 1, \\ p_{ij}^{t-1}, & \text{if } y_{ij}^t = -1, \\ \text{a random number}, & \text{otherwise.} \end{cases} \tag{10}$$

The choice previously achieved for the affectation of a random value in $\{-1,1\}$ for y_{ij}^{t-1}, in the case of equality between x_{ij}^{t-1}, p_{ij}^{t-1} and G_j^{t-1}, might insure that the variable y_{ij}^t takes a value 0, and permit to change the value of variable x_{ij}^t.

We define a parameter α as parameter for intensification and diversification. For a small value of α, x_{ij}^t takes one of the two values p_{ij}^{t-1} or G_j^{t-1} (intensification). In the opposite case, we impose to the algorithm to assign a null value to the y_{ij}^t, which induces to choose another value different from p_{ij}^{t-1} and G_j^{t-1} (diversification).

The parameters c_1 and c_2 are two parameters relative to the importance of the solution p_{ij}^{t-1} and G_j^{t-1} for the generation of the new solution X_i^t. They also have a role in the intensification of the research.

5. CPSO algorithm for solving MRCPSP

As we can see, the MRCPSP consists of two different sub-problems: the assignment of modes to tasks and then the scheduling of these tasks in order to minimize the makespan of the project.

The combinatorial PSO will deal with the first problem to generate an assignment of modes to activities which is called particle. A local search will optimize the sequences when a new assignment is made.

In this section, we take advantage of the CPSO to solve the MRCPSP. Clearly, the discrete particle needs to be redesigned to represent an assignment of J tasks to m modes. From the beginning, to each particle, we associate a sequence generated in a probabilistic way. While the particle will be optimized by the CPSO, the sequence, which will follow its particle, will be optimized, during the evolution of the particle, by the local search optimization.

To illustrate the concepts of adopting CPSO to our problem, we will use the example of assigning six tasks to three modes.

5.1. Particle presentation

In PSO, each particle corresponds to a candidate solution of the underlying problem. Thus, we let each particle represent a decision for task assignment using a vector of J elements, and each element is an integer value between 1 and m. Fig. 1 shows an illustrative example for the ith particle, in the tth iteration, which corresponds to a task assignment vector X_i^t composed of six elements x_{ij}^t to three modes. $particle_{i4}^t = x_{i4}^t = 2$ means that task "4" is assigned to mode "2" in the ith particle.

Particle i

Tasks j	1	2	3	4	5	6
X_i^t = Modes assignment	1	1	3	2	3	1

Fig. 1. Example of particle.

5.2. Initial swarm generation

The PSO randomly generates an initial swarm of K particles, where K is the swarm size.
We associate to the mode m assigned to task j the following probability p_{jm} to be selected:

$$p_{jm} = \frac{\frac{1}{\sum_{k=1}^{R_k} r_{jmk} \times d_{jm}}}{\sum_{m'=1}^{M_j} \frac{1}{\sum_{k=1}^{R_k} r_{jm'k} \times d_{jm'}}}. \tag{11}$$

The lower is the renewable resource consumption of the mode m, the higher is its associated probability and therefore, it has more chance to be selected.

Tasks j	***1***	***2***	***3***	***4***	***5***	***6***
X_i^{t-1}	1	1	3	2	3	1
P_i^{t-1}	1	2	1	3	2	1
G^{t-1}	2	2	3	1	3	1

We obtain the vector value Y_i^{t-1} (see (5)):

Y_i^{t-1}	-1	0	1	0	1	-1

Then we compute the vector value of velocity V_i^t (see (7)):

V_i^t	0.2	0.5	-0.9	-0.7	-0.4	1.2

Thus we obtain the vector value of $\lambda_{ij}^t = y_{ij}^t + v_{ij}^t$ (see (8)):

λ_i^t	-0.8	0.5	0.1	-0.7	0.6	0.2

We compare the new value of λ_i^t with the value of ; we obtain then the new vector of Y_i^t.

Let α=0.3, then:

Y_i^t	-1	1	0	-1	1	0

The new particle is then as follows:

X_i^t	1	2	1	3	3	1

Fig. 2. New solution generation.

These particle vectors will be iteratively modified based on self and collective experiences in order to improve their solution quality.

5.3. New solution generation

We obtain the vector value Y_i^{t-1} by comparing X_i^{t-1} with P_i^{t-1} and G^{t-1} vectors, as we can see in Fig. 2. If we take the example of task "3", $x_{i3}^{t-1} = G_3^{t-1} = 3$ so, from Eq. (5), we deduce that $y_{i3}^{t-1} = 1$. Then, we compute the v_{i3}^t by means of Eq. (7) where r_1 and r_2 are random real numbers drawn from $U(0,1)$. From Eq. (8) we deduce the vector value of $\lambda_{i3}^t = y_{i3}^{t-1} + v_{i3}^t = 1 + (-0.9) = 0.1$. While 0.1 is neither more than 0.3 (the value of α) nor less than -0.3 (the value of "$-\alpha$") (see Eq. (9)) then $y_{i3}^t = 0$. We deduce thereafter the new value of $x_{i3}^t = 1$, which corresponds to P_{i3}^{t-1} but x_{i3}^t could be equal to $G_3^{t-1} = 3$, because the choice is random when y_{i3}^t is null.

We apply the local search to optimize the sequence associated to the new particle X_i^t. We compute then its *fitness*.

5.4. Fitness evaluation

Each particle vector in the swarm is assigned a *fitness* value indicating the merit of this particle vector such that the swarm evolution is navigated by best particles. Clearly, the objective value of each sequence $C_{\max}$ can be used to measure the quality of the associated particle vector. This objective value represents the makespan of the sequence which results from the local search algorithm. However, infeasible solutions also provide valuable clue to targeting the optimal solution. The degree of infeasibility of solutions is measured and transformed to a penalty which grows in proportion to the infeasibility level, thus guiding the search toward the feasible space. Consequently, we devise a penalty function to estimate the infeasibility level of a particle solution.

$$\text{Penalty} = \sum_{l}^{N_l} \partial \max\left(0, \sum_{j=1}^{J} n_{jm_{(j)}l} - N_l\right). \tag{12}$$

Note that $m_{(j)}$ is the mode selected to the task j in the current solution.

$$\textit{Fitness} = C_{\max} + \textit{Penalty}. \tag{13}$$

6. Local search optimization

Starting from an arbitrary initial solution, this method seeks to obtain better solutions, by carrying out series of local modifications, which leads to a monotone decrease of the objective function. This procedure permits the scheduling of tasks in order to optimize the sequence associated to each particle. The initial sequence associated to each particle at time 0 is randomly generated. In fact, to each task j we assign the following probability p_j:

$$p_j = \frac{succ_j}{\sum_j succ_j}, \tag{14}$$

where $succ_j$ is the number of successors of this task.

The task which has the higher probability, that is to say the task with more successors, will be selected. During the optimization path of the sequence, we generate its neighbours by moving from the current solution to another as follows: a move requires two adjacent activities in positions k and $k+1$ $(j_{(k)}, j_{(k+1)})$ to be swapped in the current list. Only precedence feasible moves are considered. More precisely, activities $j_{(k)}$ and $j_{(k+1)}$ are eligible for swapping if the following condition holds: activity $j_{(k)}$ is not the predecessor of activity $j_{(k+1)}$. This condition ensures that the new list is feasible.

The local search optimization is used unless this condition is satisfied: $F_i < F_{\text{best}} \times (1 + \Delta)$, where F_i is the *fitness* of the new solution, F_{best} is the best evaluation of the objective function and Δ is a constant; this limits the feasible space.

7. Implementation and experimental results

The proposed CPSO algorithm for the MRCPSP problem was implemented in C++ programming language on Dell desktop PC with Intel Pentium 4 and 3.2 GHz processors.

In 2003, Bouleimen and Lecocq [8] and more recently, in 2006, Zhang et al. [16] have solved the MRCPSP with the simulated annealing algorithm and the PSO algorithm respectively, implemented on Pentium 100 MHz processors. To make a fair comparison with their results, we must compute our CPU time with respect to our processor capacity. Since we used a machine with 3.2 GHz, which is approximately 32 times faster than the one used by Bouleimen and Lecocq [8], we divided our CPU time by 32. For problems with $n \leqslant 20$, Bouleimen and Lecocq [8] limited their CPU time to 5 s and 10 s for the problem with 30 tasks. We fixed our time limit only to 150 ms and 300 ms, in the first and the second case, respectively.

We used a set of standard test problems systematically constructed by the project generator ProGen which has been developed by Kolisch et al. [18]. They are available in the Project scheduling Problem Library PSPLIB from the University of Kiel.

7.1. Parameter setting

From the experiments, we have fixed the parameters values as follows: the acceleration constants c_1 and c_2 take 0.8 and 0.2, respectively, the inertia weight ω, which controls the impact of the previous history of velocities on the current one, takes 0.75 and the parameter of intensification and diversification α is fixed to 0.3. Nevertheless, r_1 and r_2 are random real numbers drawn from $U(0,1)$.

In Eq. (12) ∂ should be greater than the maximal $C_{\max}$ of all solutions in order to inflate the value of the *fitness*, in the case of infeasible solutions, which induces their elimination.

We have fixed the constant Δ associated to the fitness to 0.05 and the swarm size to 200 particles.

7.2. Implementation and results

We used benchmark sets for problems with 10, 12, 14, 16, 18 and 20 non-dummy activities for which only the set of the best known heuristic solutions is available. 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. We have two renewable and two non-renewable resources. For each problem size, a set of instances was generated by systematically varying four parameters, that is, the resource factor and the resource strength of each resource category. The resource factor is a measure of the average portion of resources requested per task. The resource strength reflects the scarceness of the resources.

For each problem size, a set of 640 instances are generated but, for some instances, no feasible solution exists, therefore these instances are excluded from consideration. Hence, we have 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, cf. Sprecher and Drexl [3]. For the set with 30 activities, we have only 540 feasible instances and not all optimal solutions are known, so we compare the results with the best found until now.

The results of the experiments are presented in Table 1. For each problem size we indicate the percentage of problems solved to optimality, the total deviation, the maximal deviation and the CPU time in seconds.

It is worth noting that our CPSO is very efficient since it solved the first three problems to optimality. Even the remaining problems, except for the problem with 30 tasks, have been well solved and their results may be further improved, if we accept more execution time. Moreover, our CPU time is not bad but with more sophisticated computers it could be better.

By comparison with the results of Bouleimen and Lecocq [8], the CPSO algorithm outperforms the simulated annealing for all problems sizes. The results of our CPSO algorithm are close to those of the PSO algorithm of Zhang et al. [16].

Table 1
Solutions for MRCPSP benchmark instances

Tasks number	Feasible solutions	Simulated annealing		PSO		Proposed CPSO	
		Average deviation	Percentage of optima found	Average deviation (%)	Percentage of optima found (%)	Average deviation	Percentage of optima found
10	536	0.21	96.3	0.11	97.9	0.03	99.25
12	547	0.19	91.2	0.17	95.2	0.09	98.47
14	551	0.92	82.6	0.41	89.3	0.36	91.11
16	550	1.43	72.8	0.83	85.6	0.44	85.91
18	552	1.85	69.4	1.33	74.5	0.89	79.89
20	554	2.10	66.9	1.79	69.1	1.10	74.19
30	540					2.35	57.41

8. Conclusion

To conclude, we recall that, in this work, we investigated the multi-mode resource-constrained project scheduling problem. Since our problem is NP-Hard and composed of more than 20 tasks, we have adopted the particle swarm optimization (PSO) to solve it, our objective was the minimization of the makespan. For this purpose, we have proposed a combinatorial particle swarm optimization, that we have called CPSO. This procedure has been fruitful in the assignment of different modes to the activities. We have also used a local search method to optimize the sequence associated to each assignment. Our results proved to be very competitive in terms of percentage of problems solved to optimality and average deviations. Indeed, we have solved all the feasible instances to optimality in the three first cases ($n = 10$, 12 and 14) but in the four remaining cases, the deviations have not been null; even though they are not important.

Finally, we have presented a new method to solve the MRCPSP which consists in applying a new CPSO algorithm to assign modes to activities and local search optimization to optimize sequences associated to assignments during the evolution of the CPSO algorithm. To validate our new method we have compared it with the PSO algorithm proposed by Zhang et al. [16] and the simulated annealing algorithm of Bouleimen and Lecocq [8]. It is worth noting that our results are better for all problem sizes.

References

[1] S.E. Elmaghraby, Activity Networks: Project Planning and Control by Network Models, Wiley, New York, 1977.
[2] A. Sprecher, in: Resource-Constrained Project Scheduling: Exact Methods for the Multi-Mode Case, Lecture Notes in Economics and Mathematical Systems, vol. 409, Springer, Berlin, Germany, 1994.
[3] A. Sprecher, A. Drexl, Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm, European Journal of Operational Research 107 (1998) 431–450.
[4] A. Drexl, J. Grünewald, Nonpreemptive multi-mode resource-constrained project scheduling, IIE Transactions 25 (1993) 74–81.
[5] R. Slowinski, B. Soniewicki, J. Weglarz, DSS for multiobjective project scheduling subject to multiple-category resource constraints, European Journal of Operational Research 79 (1994) 220–229.
[6] R. Kolisch, A. Drexl, Local search for non-preemptive multi-mode resource-constrained project scheduling, IIE Transactions 29 (1997) 987–999.
[7] L. Özdamar, A genetic algorithm approach to a general category project scheduling problem, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 29 (1999) 44–59.
[8] K. Bouleimen, H. Lecocq, A new efficient simulated annealing algorithm for the resource constrained project scheduling problem, in: G. Barbarosöglu, S. Karabatı, L. Özdamar, G. Ulusoy (Eds.), Proceedings of the Sixth International Workshop on Project Management and Scheduling, Bögaziçi University Printing Office, Istanbul, 1998, pp. 19–22.
[9] S. Hartmann, A. Drexl, Project scheduling with multiple modes: a comparison of exact algorithms, Networks 32 (1998) 283–297.
[10] F.F. Boctor, Heuristics for scheduling projects with resource restrictions and several resource duration modes, International Journal of Production Research 31 (1993) 2547–2558.
[11] F.F. Boctor, A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes, European Journal of Operational Research 90 (1996) 349–361.
[12] M.F. Tasgetiren, Y.C. Liang, M. Sevkli, G. Gencyilmaz, Particle swarm optimization algorithm for makespan and maximum lateness minimization in permutation flowshop sequencing problem. In: Proceedings of the Fourth International Symposium on Intelligent Manufacturing Systems, Turkey, Sakarya, 2004, pp. 431–441.

[13] MF. Tasgetiren, YC. Liang, M. Sevkli, G. Gencyilmaz, Particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem, European Journal of Operational Research 177 (3) (2007) 1930–1947.
[14] C. Liao, C. Tseng, P. Luarn, A discrete version of particle swarm optimization for flowshop scheduling problems, Computers & Operations Research (2007) 3099–3111.
[15] A. Salman, I. Ahmad, S. Al-Madani, Particle swarm optimization for task assignment problem, Microprocessors and Microsystems 26 (2002) 363–371.
[16] H. Zhang, C.M. Tam, H. Li, Mulitmode project scheduling based on particle swarm optimization, Computer-Aided Civil and Infrastructure Engineering 21 (2006) 93–103.
[17] J. Kennedy, RC. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, 1995, pp. 1942–1948.
[18] R. Kolisch, A. Sprecher, A. Drexl, Characterization and generation of a general class of resource constrained project scheduling problems, Management Science 41 (1995) 1693–1703.