Computers & Operations Research 39 (2012) 449–460

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem

Ling Wang *, Chen Fang

Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Available online 8 May 2011

Keywords:
Multi-mode resource-constrained project scheduling
Estimation of distribution algorithm
Probability model
Permutation based local search

ABSTRACT

In this paper, an estimation of distribution algorithm (EDA) is proposed to solve the multi-mode resource-constrained project scheduling problem (MRCPSP). In the EDA, the individuals are encoded based on the activity-mode list (AML) and decoded by the multi-mode serial schedule generation scheme (MSSGS), and a novel probability model and an updating mechanism are proposed for well sampling the promising searching region. To further improve the searching quality, a multi-mode forward backward iteration (MFBI) and a multi-mode permutation based local search method (MPBLS) are proposed and incorporated into the EDA based search framework to enhance the exploitation ability. Based on the design-of-experiment (DOE) test, suitable parameter combinations are determined and some guidelines are provided to set the parameters. Simulation results based on a set of benchmarks and comparisons with some existing algorithms demonstrate the effectiveness of the proposed EDA.

1. Introduction

The multi-mode resource-constrained project scheduling problem (MRCPSP) is an extension of the resource-constrained project scheduling problem (RCPSP), which is concerned with scheduling the project activities over time and resources. Generally, the MRCPSP is much more close to the reality. In a multi-mode resource-constrained project, each activity has several execution modes, each of which has a related duration and nonrenewable/renewable resource requirements. The MRCPSP is more complex than the RCPSP. Kolisch and Drexl [1] have proven that even finding a feasible solution of the MRCPSP with more than one nonrenewable resource is NP-complete. Until now, only the MRCPSP with no more than 20 activities could be solved optimally using exact algorithms with acceptable computational time [2]. In practice, the MRCPSP usually should be solved in limited time. As a result, heuristics for the MRCPSP have gained increasing attention during the past decades.

Kolisch and Drexl [1] proposed a three phase local search approach: construction phase to generate the initial solution, local search phase to perform a neighborhood search on the set of feasible mode assignment, and intensification phase to generate a schedule with a near-optimal heuristic based on the mode assignment given by the local search phase. Hartmann [3] developed a genetic algorithm (GA) to solve the MRCPSP, in which single-pass and multi-pass local search component were adopted as local search to improve the schedules. Hartmann also defined a unique similarity measure to analyze the similarity of the whole population, with which it concluded that the GA with inheritance mechanism deteriorated the quality of the solution in a long-term evolution. Józefowska et al. [4] proposed a simulated annealing (SA) algorithm to solve the MRCPSP, in which two versions of SA were discussed: SA without penalty function and SA with penalty function. Bouleimen and Lecocq [5] also proposed a simulated annealing algorithm for the MRCPSP, in which an approach was introduced to use two embedded search loops alternating activity and mode neighborhood exploration to improve the search. Alcaraz et al. [6] also proposed a GA to solve the MRCPSP. Different from the GA in [3], sophisticated designed solution representation and fitness function were incorporated to improve the performance. Zhang and Tam [7] proposed a particle swarm optimization (PSO) based approach, in which a procedure checking and adjusting infeasible particle-represented solutions was adopted to transform infeasible solutions into feasible ones. Jarboui et al. [8] proposed a combinatorial PSO approach, in which a unique representation and a local search procedure were adopted. Van Peteghem and Vanhoucke [9] proposed an artificial immune system (AIS) to solve the MRCPSP. For the initial population, the mode assignment procedure was translated to multi-choice multi-dimensional knapsack problem and the activity list was generated with priority rules. Wauters et al. [10] proposed a multi-agent learning approach, in which an agent

* Corresponding author. Tel.: +86 10 62783125; fax: +86 10 62786911.
E-mail address: wangling@mail.tsinghua.edu.cn (L. Wang).

0305-0548/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2011.05.008

represented an activity node and for each node the related agent decided how to choose the mode of the activity and how to visit its successors. Damak et al. [11] proposed a differential evolution (DE) approach to solve the MRCPSP, and the effect of population size on the solution quality was analyzed. Elloumi and Fortemps [12] proposed a hybrid rank-based evolutionary algorithm that transformed the MRCPSP to a bi-objective problem with the objectives of makespan and nonrenewable resource consumption. As a result, the nonrenewable resource constraints were relaxed and the evolutionary operators were simplified. Ranjbar et al. [13] proposed a hybrid scatter search to solve the discrete time/resource trade-off problem and the MRCPSP. Tseng and Chen [14] proposed a two-phase genetic local search algorithm that contained two phases: the first phase aimed at searching globally for promising areas while the second phase aimed at searching in the promising areas more thoroughly. Lova et al. [15] proposed an efficient hybrid GA to solve the MRCPSP, in which a unique mode assignment procedure was developed and a new fitness function was proposed to improve the results. Van Peteghem and Vanhoucke [16] proposed a bi-population GA to solve preemptive and non-preemptive MRCPSP that made use of two separate populations and extended the serial schedule generation scheme by introducing a mode improvement procedure.

Estimation of distribution algorithm (EDA) is a kind of stochastic optimization algorithm based on statistical learning [17]. Unlike GA explicitly applies crossover and mutation operator to generate new individuals, EDA generates new individuals by predicting the most promising area based on the distribution of elite individuals of former generations in the search space. Refer [17] for more details about EDA. So far, EDA has been developed to solve a variety of optimization problems in academic and engineering fields, such as feature selection [18], flow-shop scheduling [19], nurse rostering [20], quadratic assignment problem [21], multi-speed planetary transmission design [22], inexact graph matching [23], and software testing [24]. In this paper, we propose an EDA for solving the MRCPSP with the criterion to minimize makespan. In particular, the activity-mode list (AML) is adopted as the encoding scheme, and a novel probability model and updating mechanism are developed to help identify the most promising area, and a multi-mode forward backward iteration (MFBI) and a multi-mode permutation based local search method (MPBLS) are applied to the best individuals to exploit the neighborhood of the best individuals. We investigate the parameter setting of the proposed EDA based on DOE tests and carry out simulation test based on a set of benchmarks. Computational results and comparisons demonstrate the effectiveness of the EDA.

The remainder of the paper is organized as follows: In Section 2, the MRCPSP is described. In Section 3, the basic EDA is introduced. Then, the EDA for the MRCPSP is proposed in Section 4. Computational results and comparisons are provided in Section 5. Finally we end the paper with some conclusions in Section 6.

2. Multi-mode resource-constrained project scheduling problem

The MRCPSP is to study how to allocate renewable/nonrenewable resource and schedule activities to minimize the whole project makespan. Generally, the MRCPSP can be stated as follows. A project is composed of J activities, and each activity has M execution modes. The set of renewable resource is referred as K^{ρ}, and the per-period-availability is assumed to be constant R_k^{ρ}. The set of nonrenewable resource is referred as K^{v}, the total amount of availability is R_k^{v}. Precedence relationship for each activity $j=1,2,\ldots,J$ is defined by sets of its immediate predecessors P_j, which indicate that the activity j cannot be executed before any of its predecessors $i \in P_j$ is not completed yet. If activity $j=1,2,\ldots,J$ is executed in mode $m \in \{1,\ldots,M_j\}$, the duration is d_{jm}. In this duration, r_{jmk} units of renewable resource $k \in K^{\rho}$ is consumed in each period of the non-preemptable duration, and n_{jmk} units of nonrenewable $k \in K^{v}$ is consumed totally. The dummy activities $j=0$ and $j=J+1$ represent the start and end of the project, respectively. The dummy activities do not request any resource and have zero duration. The set of all activities can be denoted as $J^{+}=\{0,\ldots,J+1\}$. The objective is to minimize the makespan of the project. The MRCPSP can be formulated as follows:

$$\text{Minimize} \quad \sum_{t=EF_{J+1}}^{LF_{J+1}} t x_{J+1,1,t} \tag{1}$$

subject to

$$\sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} x_{jmt} = 1, \quad j \in J^{+} \tag{2}$$

$$\sum_{m=1}^{M_i} \sum_{t=EF_i}^{LF_i} t x_{imt} \le \sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} (t-d_j) x_{jmt}, j \in J^{+}; i \in P_j \tag{3}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} r_{jmk} \sum_{b=\max\{t,EF_j\}}^{\min\{t+d_{jm}-1,LF_j\}} x_{jmb} \le R_k^{\rho}, \quad k \in K^{\rho} \tag{4}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} n_{jmk} \sum_{t=EF_j}^{LF_j} x_{jmt} \le R_k^{v}, \quad k \in K^{v} \tag{5}$$

$$x_{jmt} = \begin{cases} 1, & \text{if activity } j \text{ is performed in mode } m \text{ and finished at time } t \\ 0, & \text{otherwise} \end{cases}, \quad j \in J^{+} \tag{6}$$

where Eq. (1) defines the objective to minimize the makespan; Eq. (2) guarantees that each activity is executed exactly once; Eq. (3) confirms the precedence relationship to be satisfied; Eqs. (4) and (5) represent the renewable and nonrenewable resource restriction, respectively.

Fig. 1 illustrates a simple example with 11 activities (including two dummy activities $j=0$ and $j=J+1$), where each activity has two modes. In this example, one kind of renewable resource and one kind of nonrenewable resource are considered, and the available amounts of available renewable and nonrenewable resource are 2 and 28, respectively. The resource requirements and corresponding durations for the two modes are listed in Table 1.

In Fig. 2, a schedule of this project is shown. The mode of each activity $j=1,2,\ldots,J$ is 2, 1, 2, 1, 2, 2, 2, 2, 1, respectively. Since both precedence constraint and the amounts of nonrenewable resource consumed are 25 (less than 28), it is a feasible schedule with makespan 18.

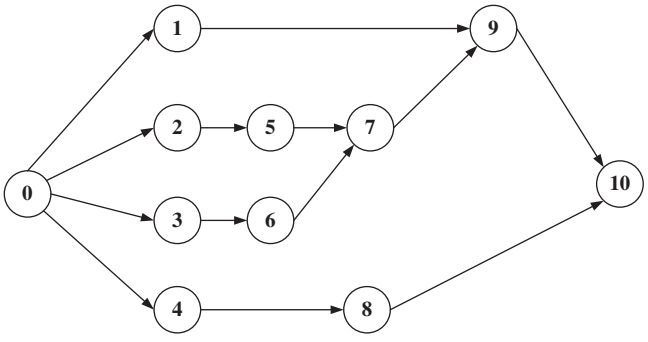


Fig. 1. An example of the MRCPSP.

3. Estimation of distribution algorithm

Estimation of distribution algorithm (EDA) can be regarded as a general framework of statistical learning based optimization algorithms. With the tool of statistical analysis, the EDA tries to predict of the movement of population in the search space and estimates the underlying probability distribution of encoded variables of the elite individuals. After generating the initial population and initializing the probability matrix, an iterative procedure is carried out to estimate the distribution of the optimal solutions until the stopping condition is met. The general framework of the EDA is illustrated in Fig. 3.

The core of the EDA procedure is to estimate the probability distribution. Due to the difference of problem types, different probability models can be designed to estimate the underlying probability distribution. In the next section, we will propose a special probability model and an updating mechanism based on the searching mechanism of the EDA for solving the MRCPSP.

4. The EDA for MRCPSP

In this section, we first introduce the encoding scheme, probability model, probability generating mechanism, local search strategy, and updating mechanism. Then, we present the framework of the proposed EDA.

4.1. Preprocessing

Before starting the EDA, we utilize the reduction procedure developed by Sprecher et al. [25] to reduce the search space. This procedure could exclude the inefficient/non-executable mode as well as redundant resources. For integrity, we briefly introduce the definitions as follows:

Inefficient mode: A mode m' is called inefficient if there exists another mode m with $d_{jm} \le d_{jm'}$ and $r_{jmk} \le r_{jm'k}$ for each renewable resource $k \in K^{\rho}$ and $n_{jmk} \le n_{jm'k}$ for each nonrenewable resource $k \in K^{v}$.

Table 1
Resource requirements and corresponding durations.

Activity	Mode 1			Mode 2		
	r_{jm_jk}	n_{jm_jk}	d_{jm_j}	r_{jm_jk}	n_{jm_jk}	d_{jm_j}
1	2	4	2	1	2	3
2	1	5	2	1	2	4
3	1	3	2	1	1	3
4	1	2	1	1	1	2
5	2	5	2	2	3	4
6	1	4	2	1	3	3
7	2	2	1	2	1	2
8	2	3	1	1	2	3
9	1	3	4	1	1	6

Non-executable mode: A mode m' is called non-executable if $\sum_{i=1,i\neq j}^{J} \min_{m} n_{imk} + n_{jm'k} > R_k^v$.

Redundant resource: A nonrenewable resource $k \in K^{v}$ is called redundant if $\sum_{j=1}^{J} \max_{m} n_{jmk} \le R_k^v$. The pseudo code of the preprocessing is summarized in Fig. 4.

4.2. Encoding scheme

In our EDA, we adopt the activity-mode list (AML) based encoding scheme, which contains two parts: (1) an activity list (AL) $\{\pi_0,\pi_1,\pi_2,\ldots,\pi_{J+1}\}$; (2) a mode assignment list (ML) $\{\sigma_0,\sigma_1,\sigma_2,\ldots,\sigma_{J+1}\}$. The σ_j in the ML indicates the mode of the π_j in the AL. Kolisch and Hartmann [26] concluded by

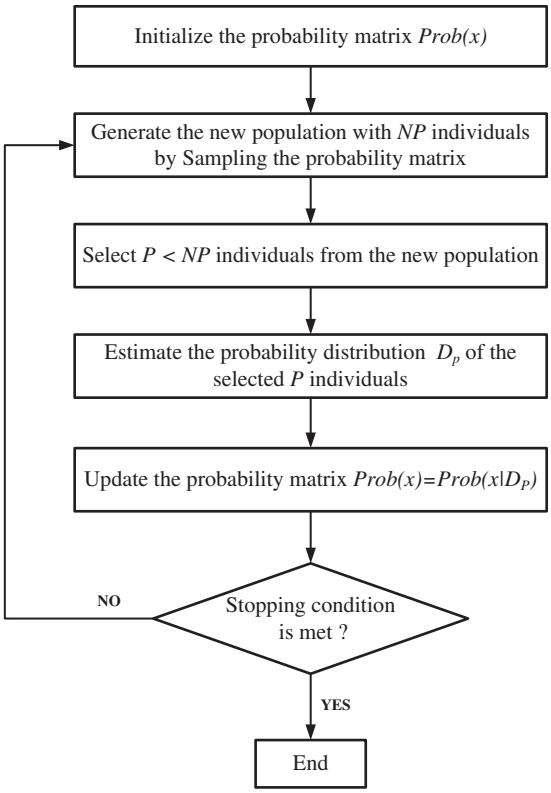


Fig. 3. The general framework of EDA.

Step1: Remove all the non-executable modes;
Step2: Delete the redundant nonrenewable resources;
Step3: Eliminate all the inefficient modes;
Step4: If any mode has been erased within Step 3, go to Step2.

Fig. 4. Procedure of preprocessing.

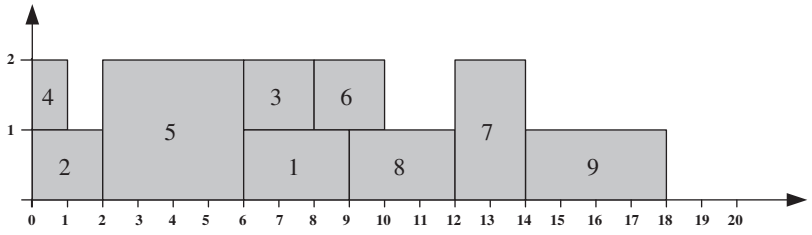
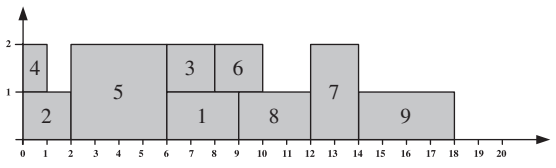


Fig. 2. A feasible schedule of the example in Fig. 1.



AL	2	4	5	1	3	6	8	7	9
ML	1	1	2	2	2	2	2	2	1

Fig. 5. The AML representation for the schedule shown in Fig. 2.

experimental tests that procedures based on AL representations outperformed other procedures. In the MRCPSP, the mode assignment list is needed to assign every activity a mode. The EDA does not operate on a schedule but on the AML representation of a schedule. This character makes it convenient and effective for solving the MRCPSP.

With the above encoding scheme, a multi-mode schedule generation scheme (MSSGS) is used to transform the representation to a schedule. When a new AML is generated (the AL and ML part of the AML is changed by searching operation), the MSSGS is used to evaluate the AML. Consider the simple instance illustrated in Fig. 1, the AML representation for the schedule (as Fig. 2) is shown in Fig. 5. In the next section, we will present the multi-mode serial SGS (MSSGS) to implement the searching operations effectively in the EDA based on the AML.

4.3. Multi-mode serial schedule generation scheme

Schedule generation scheme (SGS) is an effective heuristic procedure that is used to decode solution representation of the RCPSP to a schedule [27]. Owing to the SGS, many algorithms could solve the RCPSP conveniently. However, it is more complicated to transform a solution representation of the MRCPSP to a schedule, since there are two more tough things that need to consider. First, there exists the nonrenewable resources infeasible solution representation that cannot be transformed into a schedule. Second, for the nonrenewable resources feasible representation, the nonrenewable resources may not be the utilized fully.

So far, many fitness functions have been proposed in literature for solving the MRCPSP, such as Hartmann [3], Alcaraz et al. [6], Jarboui et al. [8] and Lova et al. [15]. According to Lova et al. [15], their fitness function gave better results than other ones over all instances of PSPLIB. So, we adopt the fitness function proposed by Lova et al.. For an AML with ML $\{\sigma_0,\sigma_1,\sigma_2,\ldots,\sigma_{J+1}\}$, the fitness function is computed as follows:

$$f(AML)=\begin{cases}1-\frac{max\_mak-mak(AML)}{max\_mak} & \text{if} \quad ERR(ML)=0\\ 1+\frac{mak(AML)-min\_CP}{mak(AML)}+ERR(ML) & \text{otherwise}\end{cases} \tag{7}$$

where $mak(AML)$ is the makespan of AML; max_mak is the maximal makespan of feasible solutions related to individuals of the current generation; min_CP is the critical path using the minimal duration of each activity; and $ERR(ML)$ is the nonrenewable infeasible degree of ML, which is calculated as follows:

$$ERR(ML)=\sum_{k=1}^{K^v}\max\left\{0,\frac{\sum_{j=1}^{J}n_{j\sigma_j k}-R_k^v}{R_k^v}\right\} \tag{8}$$

It is noticeable that $ERR(ML)=0$ if the AML is nonrenewable resources feasible.

Based on that fitness function, we propose a multi-mode serial schedule generation scheme (MSSGS) for the MRCPSP. The MSSGS contains two procedures: infeasible tackling procedure to tackle infeasible representation and feasible tackling procedure to tackle feasible representation.

In order to transform a nonrenewable resource infeasible solution into a feasible one, we adopt the initial population local search procedure developed by Hartmann [3] as the infeasible tackling procedure, which chooses an activity randomly and changes its mode to a new one. If the *ERR* is reduced, the new mode is kept. Such a procedure is repeated until the representation becomes nonrenewable resource feasible or the maximum iteration number *SGS_adj* is reached. If the solution is still infeasible, we calculate its fitness value. In our procedure, we adopt the number of activities J as the maximum iteration number. That is, $SGS\_adj=J$.

In order to fully use the nonrenewable resources, the feasible tackling procedure is developed, which is based on the multi-mode left shift bounding rule developed by Sprecher et al. [25]. The multi-mode left shift is a rule to improve the finishing time of activity one by one at the every decision point of the partial schedule by changing its mode without changing the modes and delaying the finishing times of any other activities. This rule could help utilize the nonrenewable resources more effectively. However, there is a weakness of the multi-mode left shift bounding rule. The activities scheduled earlier will occupy more amounts of nonrenewable resources. Whereas, the activities scheduled later will not have enough amounts of nonrenewable resources to use. As a result, this solution may trap into local minima. In our procedure, we develop a probability based selecting scheme controlled by a threshold *SGS_Pper* to guarantee that all the activities have equal opportunity to use nonrenewable resources.

The pseudo code of the MSSGS is illustrated in Fig. 6.

In a word, the MSSGS can reduce the nonrenewable resource consumption for infeasible representations to transform it into a feasible one and it also can increase the nonrenewable resource consumption to improve the makespan for feasible representations.

4.4. Probability model

Different from the GA that produces offspring through crossover and mutation operators, the EDA does it by sampling according to a probability model, which has a great effect on the performance. How to construct the probability model is the key issue to design the EDA [28].

In this paper, the probability model is composed of two probability matrixes: a $J\times J$ probability matrix $Prob\_act(t)$, and a $J\times M$ probability matrix $Prob\_mod(t)$.

(1) $Prob\_act(t)$: this matrix is used to predict the position of each activity in the *AL*.

$$Prob\_act(t)=\begin{pmatrix}\lambda_{11} & \cdots & \lambda_{1J}\\ \vdots & \ddots & \vdots\\ \lambda_{J1} & \cdots & \lambda_{JJ}\end{pmatrix} \tag{9}$$

where the element λ_{ji} represents the probability that the activity j is placed at position i of the *AL* at generation t. That is, λ_{ji} is an index to indicate how good it seems to place activity j at position i.

This probability matrix is initialized as follows to ensure that the whole solution space can be sampled uniformly.

$$Prob\_act(0)=\begin{pmatrix}\frac{1}{J} & \cdots & \frac{1}{J}\\ \vdots & \ddots & \vdots\\ \frac{1}{J} & \cdots & \frac{1}{J}\end{pmatrix} \tag{10}$$

Procedure MSSGS (AML)

Initialization j=1, $\pi R_{kt} = R_k^{\rho}$;

BEGIN

Calculate the ERR (ML);

IF ERR (ML)>0

DO

Randomly select an activity j;

Select a new mode for activity j, create a new mode assignment list ML'

IF ERR (ML') < ERR (ML)

$ML = ML'$;

k++;

WHILE k < SGS_adj && ERR (ML)==0;

IF ERR (ML)>0

$$\mathrm{f}(AML) = 1 + \frac{mak(AML) - min\_CP}{mak(AML)} + ERR(ML);$$

ELSE

WHILE j<J+1

Calculate the remaining capacity of resource r in period t $\pi R_{rt} = R_k^{\rho} - \sum_{j \in A_t} r_{j\sigma_j k}$;

Calculate the earliest precedence feasible finish time $EFT_{\pi_j} = \max\{FT_i \mid i \in P_{\pi_j}\} + d_{\pi_j\sigma_j}$;

Randomly generate q where 0<q<1;

IF q < SGS_Pper

Calculate all the finish times with different modes $m = 1, \ldots, M$:

$FT_{\pi_j m} = \min\{t \mid EFT_{\pi_j} \le t, r_{\pi_j mk} \le \pi R_{k\tau}, \tau = t - d_{\pi_j m} + 1, \ldots, t, k \in K^{\rho}\}$;

Select the mode with the minimized finish time of activity j: $\sigma_j = \arg \min_{\substack{m=1,\ldots,M_j \\ ERR(m)=0}} FT_{\pi_j m}$;

$FT_{\pi_j} = FT_{\pi_j\sigma_j}$;

ELSE

Calculate the finish time:

$FT_{\pi_j} = \min\{t \mid EFT_{\pi_j} \le t, r_{\pi_j\sigma_j k} \le \pi R_{k\tau}, \tau = t - d_{\pi_j\sigma_j} + 1, \ldots, t, k \in K^{\rho}\}$;

$ST_{\pi_j} = FT_{\pi_j} - d_{\pi_j\sigma_j}$;

j=j+1;

$$\mathrm{f}(AML) = 1 - \frac{max\_mak - mak(AML)}{max\_mak}$$

END

Fig. 6. Procedure of the MSSGS.

(2) *Prob*_mod(t): this matrix is used to predict the mode adopted by each activity.

$$Prob\_mod(t) = \begin{pmatrix} \mu_{11} & \cdots & \mu_{1M} \\ \vdots & \ddots & \vdots \\ \mu_{J1} & \cdots & \mu_{JM} \end{pmatrix} \tag{11}$$

where the element μ_{ji} represents the probability that the activity j adopt mode i at generation t. That is, μ_{ji} is an index to indicate how good it seems for activity j to be carried out in mode i.

This probability matrix is initialized as follows to ensure that the whole solution space can be sampled uniformly.

$$Prob\_\mathrm{mod}(0) = \begin{pmatrix} \frac{1}{M} & \cdots & \frac{1}{M} \\ \vdots & \ddots & \vdots \\ \frac{1}{M} & \cdots & \frac{1}{M} \end{pmatrix} \tag{12}$$

It needs to mention that μ_{ji} is set zero if the mode i of activity j is excluded as the inefficient or non-executable mode in the preprocessing procedure.

4.5. Probability generating mechanism

In order to generate a population based on the probability model, we should generate the selection probability of activity j firstly. In this paper, we introduce a permutation-based probability generating mechanism (PGM).

At every position i, the selection probability of activity j, i.e. Pa_{ji}, is calculated according to probability matrix *Prob_act*(t) over the set of eligible activities D_a, that is

$$Pa_{ji} = \frac{\lambda_{ji}}{\sum_{h=1}^{D_a} \lambda_{hi}} \tag{13}$$

Procedure MPBLS (AML)
Randomly generate q where $0<q<1$;
For (i=1,2,…,J-1)
{
  If ($q<EDA\_Pper$)
  {
    Randomly change the mode of activity j;
    If (π_i is not the predecessor for π_{i+1})
    {
      Exchange π_i and π_{i+1};
      Exchange σ_i and σ_{i+1};
    }
  }
}
Evaluation the new AML.

Fig. 7. Procedure of the MPBLS.

If activity j has already been placed in some positions, the whole line $\lambda_{j1},\lambda_{j2},\ldots,\lambda_{jJ}$ of probabilistic matrix *Prob_act* is set zero. As a result, each activity could be selected only once in an *AL*. According to selection probability of each activity, we will generate an *AL* by selecting activity one by one. Then we choose a mode for each activity in the AL according to the mode selection probability over the set of eligible modes D_m, that is

$$Pm_{ji} = \frac{\mu_{ji}}{\sum_{h=1}^{D_m} \mu_{hi}} \tag{14}$$

4.6. Local search strategy

From the mechanism of the EDA, it seems that the EDA stresses more exploration than exploitation. To enhance the exploitation ability, a multi-mode version permutation-based local search strategy (MPBLS) is proposed to explore the neighborhood of the individual. The MPBLS exploits the neighborhood of AML systematically with the following procedure described in Fig. 7.

4.7. Multi-mode forward–backward iteration

Forward–Backward Improvement (FBI) is an effective technique for the RCPSP introduced by Li and Willis [29]. Basically, the procedure iteratively employs the SGS forward and backward scheduling until no further improvement in the makespan of project can be found. With the help of the MSSGS, we could extend the classic FBI to a multi-mode version FBI (MFBI). The MFBI could improve the fitness value not only by changing the AL sequence but also by changing the modes of activities to make use of nonrenewable resources more effectively. In Fig. 8, a single iteration step is used to illustrate procedure of the MFBI.

We now try to reduce the makespan of schedule in Fig. 2 by the MFBI. First, forward iteration shifts each activity to right as much as possible in descent order of activity ending times. With the help of the MSSGS, activity 5 is changed to mode 1. As a result, its duration reduces to 2 and NRC increases from 25 to 27. In this way, we obtain a schedule with a makespan of 14 units. Second, backward iteration shift activities as much as possible to left in ascent order of activity starting times. Consequently, the makespan is reduced to 12, as illustrated in Fig. 8.

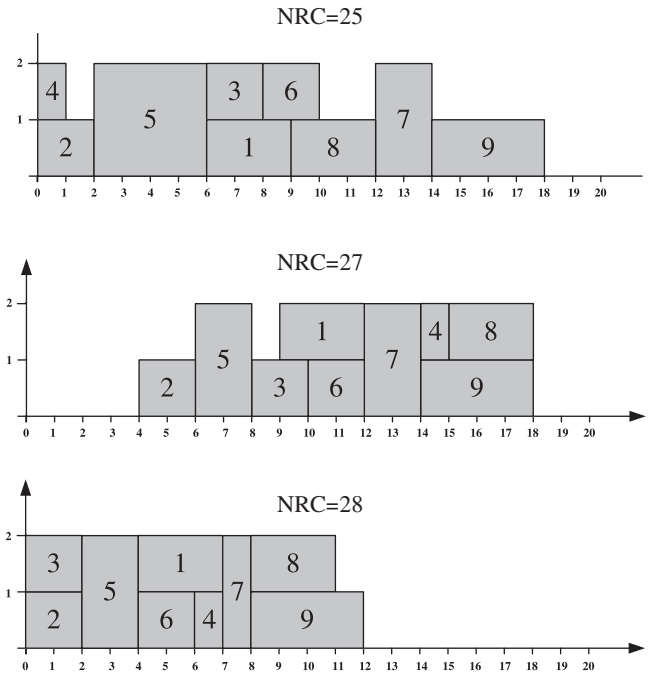


Fig. 8. A single iteration step of MFBI and nonrenewable resource consumed (NRC).

4.8. Updating mechanism

In this paper, a population based updating mechanism is proposed to update the probabilistic matrixes *Prob_act*(t) and *Prob_mod*(t). First, a number of *NP* individuals are generated according to the matrixes, and MFBI and MPBLS are employed to renew the best *P* individuals. Then, the best *P* individuals are chosen to update the probabilistic matrixes according to the following equation:

$$prob\_act_{ji}(t+1) = (1-\beta)\cdot prob\_act_{ji}(t) + \frac{\beta}{N}\sum_{k=1}^{P} I_{ji}^{k}, (1 \le i,j \le J) \tag{15}$$

$$prob\_mod_{ji}(t+1) = (1-\beta)\cdot prob\_mod_{ji}(t) + \frac{\beta}{N}\sum_{k=1}^{P} R_{ji}^{k}, (1 \le i \le M_j, 1 \le j \le J) \tag{16}$$

where β is the learning speed, I_{ji}^{k} and R_{ji}^{k} are the indicator functions.

The indicator functions are calculated as follows:

$$I_{ji}^{k} = \begin{cases} 1 & \text{if activity } j \text{ is placed at position } i \\ 0 & else \end{cases} \tag{17}$$

$$R_{ji}^{k} = \begin{cases} 1 & \text{if activity } j \text{ chooses mode } i \\ 0 & else \end{cases} \tag{18}$$

4.9. Procedure of the EDA

With the above design, the procedure of EDA for the MRCPSP is summarized as follows: First, the probability matrix is initialized uniformly. Second, the new population with *NP* individuals is generated by sampling the probability matrix using the PGM. During every generation, all the individuals are evaluated by the MSSGS and sorted in ascent order according to the makespan values. The best P ($P<NP$) individuals are selected from the population. Then the MFBI is applied to the selected *P* individuals

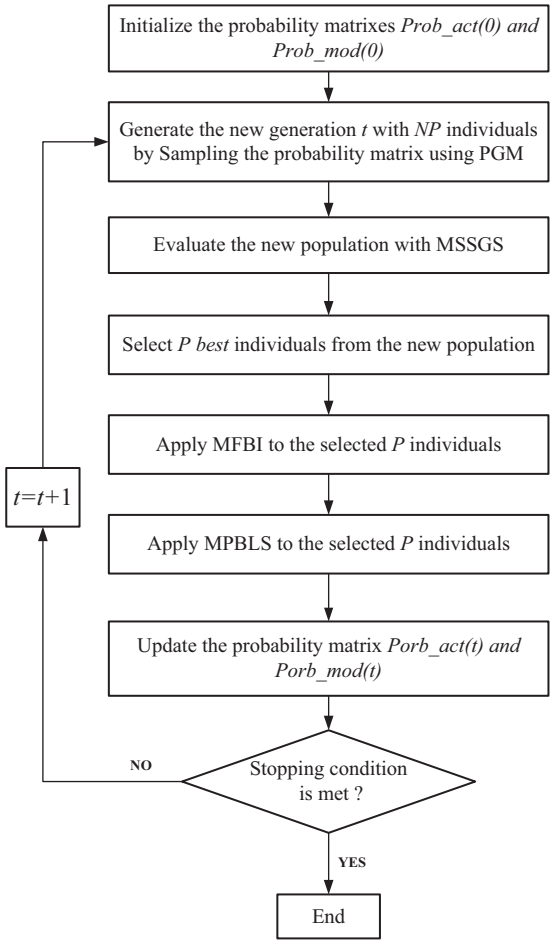


Fig. 9. The framework of the EDA for the MRCPSP.

to improve the makespan value. After that, the MPBLS is applied to them for further improvement. After the local search operation, the probability matrixes *Prob_act*(t) and *Prob_mod*(t) are updated based on the selected P individuals. Straightforwardly, the framework of the EDA is illustrated in Fig. 9.

In the next section, we will carry out experiments based on a set of benchmarks and compare the EDA with some existing algorithms.

5. Computational experiments

We code the procedure in Visual C++ 2005 on IBM Thinkpad T61 with a Core 2 T7500 2.2 GHz processor and use the standard MRCPSP test data sets J10, J12, J14, J16, J18, J20 and J30 from the well-known PSPLIB for testing, which are generated by the problem generator ProGen designed by Kolisch and Sprecher [30]. The optimal solutions for J10-20 sets are available, however no optimal solutions for J30 set were found. Each data set contains 640 instances, some of which are infeasible. We exclude the infeasible instances from our tests. Hence, we have 536 instances for J10, 547 instances for J20, 551 instances for J14, 550 instances for J16, 552 instances for J18, and 554 instances for J20. For the set J30, only 552 instances have found a feasible solution and not all optimal solutions are found. Each instance of J10-20 and J30 contains three modes (i.e. $M_j=3$, $j=1,2,\ldots J$; $M_0=1$; $M_{J+1}=1$), two renewable and two nonrenewable resources.

In the literature of the MRCPSP, two kinds of stopping conditions are used for comparison: the maximum number of the generated schedules and the maximum CPU time consumed. We adopt the following three most frequently used stopping conditions for comparison: (1) 5000 generated schedules; (2) 1 s CPU time; (3) 0.15 s CPU time per activity. According to Lova et al. [15] and Van Peteghem and Vanhoucke [16], 1 generated schedule is defined as each activity of the project has obtained exactly only one feasible start time. In our MSSGS procedure, we have a probability *SGS_Pper* for each activity to obtain the feasible start times for all the modes of this activity. In order to make a fair comparison with other algorithms available in literature, 1 schedule generated by the MSSGS is counted as

$$\frac{\sum_{j=1}^{J}(SGS\_Pper\cdot M_j+(1-SGS\_Pper))}{J}=1+2\cdot SGS\_Pper$$

generated schedules.

5.1. Parameters setting

First, we use Taguchi method of design of experiment (DOE) [31] to determine a set of suitable parameters for the EDA. Since the J20 is the hardest data set of J10-20 sets, of which the optimal solutions are known, we choose the data set J20 to carry out the DOE test.

The EDA contains five key parameters: the population size of each generation (*NP*), the number of selected individual to update the probability matrix (P), the learning speed (β), the MPBLS acceptance rate (*EDA_Pper*), and the threshold of the MSSGS (*SGS_Pper*). Combinations of different values of these parameters are shown in Table 2.

The average response variable (*ARV*) value is the following average deviation value for $N=554$ instances.

$$ARV=\sum_{i=1}^{N}\frac{(Makespan_i-OPT_i)}{OPT_i}\Big/N \tag{19}$$

where $Makespan_i$ is the makespan of the ith feasible instance in J20 obtained by the EDA; OPT_i is the optimal makespan of ith feasible instance in J20.

According to the number of parameters and the number of factor levels, we choose the orthogonal array $L_{25}(5^5)$. That is, the total number of treatments is 25, the number of parameters is 5, and the number of factor levels is 5. The orthogonal arrays for J20 are listed in Table 3.

According to the orthogonal table, we illustrate the trends of each factor level with different stopping conditions in Figs. 10–12. Then, we figure out the response value changes of each parameter to analyze the significance rank of each parameter. The corresponding results are listed in Tables 4–6.

From Tables 4–6, it can be seen that: β and *SGS_Pper* are the two parameters that have the most significant impact on the algorithm no matter which stopping condition is used. According to the factor level trends, the best combinations of parameter values for the proposed EDA are determined, which are listed in Table 7.

Table 2
Combinations of parameter values.

Parameters	Factor level				
	1	2	3	4	5
NP	50	100	150	200	250
P	2%*NP*	6%*NP*	10%*NP*	20%*NP*	30%*NP*
β	0.005	0.01	0.05	0.1	0.5
EDA_Pper	0.1	0.3	0.5	0.7	0.9
SGS_Pper	0.1	0.3	0.5	0.7	0.9

From Table 7, we can see that the best parameters combination varies with maximum CPU time (the time of 5000 schedules equals to 0.143 s). Consequently, we can see that the parameters P, β, and *SGS_Pper* decrease as the maximum CPU time increases and *NP* keeps the same all the time. We utilize the linear least square method to determine the relationship between the value of each parameter (except *NP*) and the maximum CPU time (see Fig. 13). Since the EDA is some kind of incremental learning based method, the iterative number has a great impact on the parameters selection, e.g. in order to get a better result, the learning speed with less number of generated schedules should be larger than the one with more number of generated schedules.

Clearly, different computers may have different computing power. To make sure the proposed EDA can be used on different computers, we provide the following guidelines to choose the EDA parameters based on maximum generated schedules z

$$NP = 100 \tag{20}$$

$$p = -1.256e^{-6}z + 0.1806 \tag{21}$$

Table 3
Orthogonal table for the EDA.

Experiment number	Factors					ARV		
	NP	*P*	β	*EDA_Pper*	*SGS_Pper*	5000 schedules	1 s	0.15 s/ activity
1	1	1	1	1	1	0.058022	0.036863	0.024882
2	1	2	2	2	2	0.035665	0.019301	0.005378
3	1	3	3	3	3	0.027621	0.005462	0.002766
4	1	4	4	4	4	0.023170	0.005076	0.003854
5	1	5	5	5	5	0.015185	0.008185	0.006324
6	2	1	2	3	4	0.027101	0.016161	0.010575
7	2	2	3	4	5	0.025526	0.012620	0.003707
8	2	3	4	5	1	0.048390	0.003140	0.001410
9	2	4	5	1	2	0.010394	0.005058	0.004290
10	2	5	1	2	3	0.025911	0.015230	0.009339
11	3	1	3	5	2	0.036442	0.019558	0.002041
12	3	2	4	1	3	0.026541	0.008001	0.002489
13	3	3	5	2	4	0.019360	0.005079	0.004280
14	3	4	1	3	5	0.026177	0.013500	0.009315
15	3	5	2	4	1	0.050191	0.034461	0.021878
16	4	1	4	2	5	0.027436	0.014282	0.004068
17	4	2	5	3	1	0.027473	0.003603	0.003014
18	4	3	1	4	2	0.031696	0.020565	0.013681
19	4	4	2	5	3	0.027641	0.015676	0.009568
20	4	5	3	1	4	0.025644	0.012630	0.005908
21	5	1	5	4	3	0.027905	0.003956	0.002417
22	5	2	1	5	4	0.023267	0.014251	0.009278
23	5	3	2	1	5	0.024536	0.013708	0.009215
24	5	4	3	2	1	0.048483	0.028137	0.005880
25	5	5	4	3	2	0.032129	0.012271	0.003299

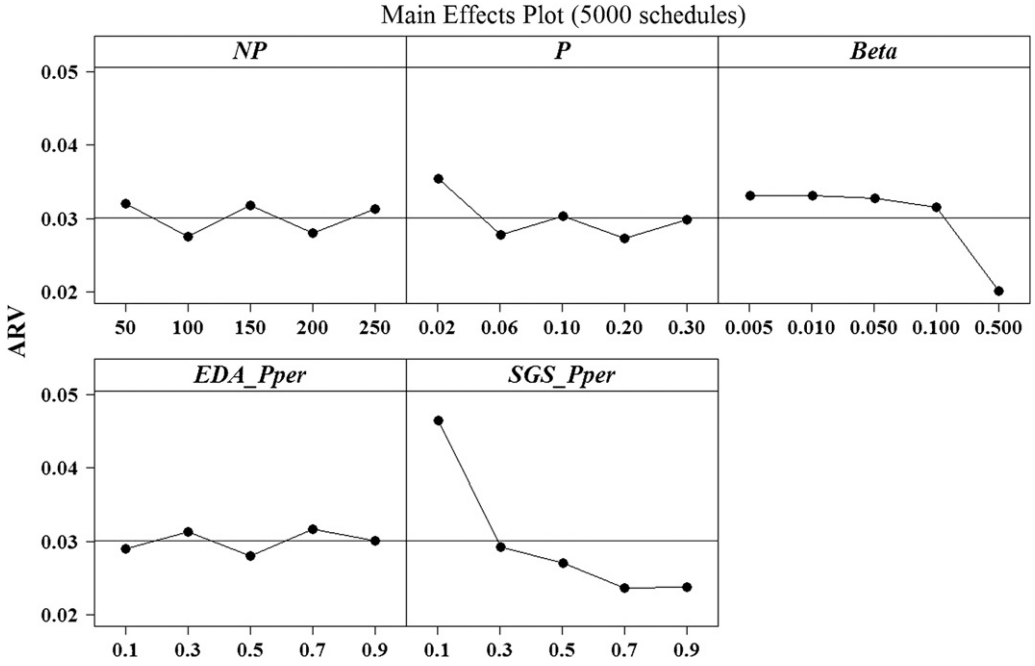


Fig. 10. Factor level trend with 5000 schedules.

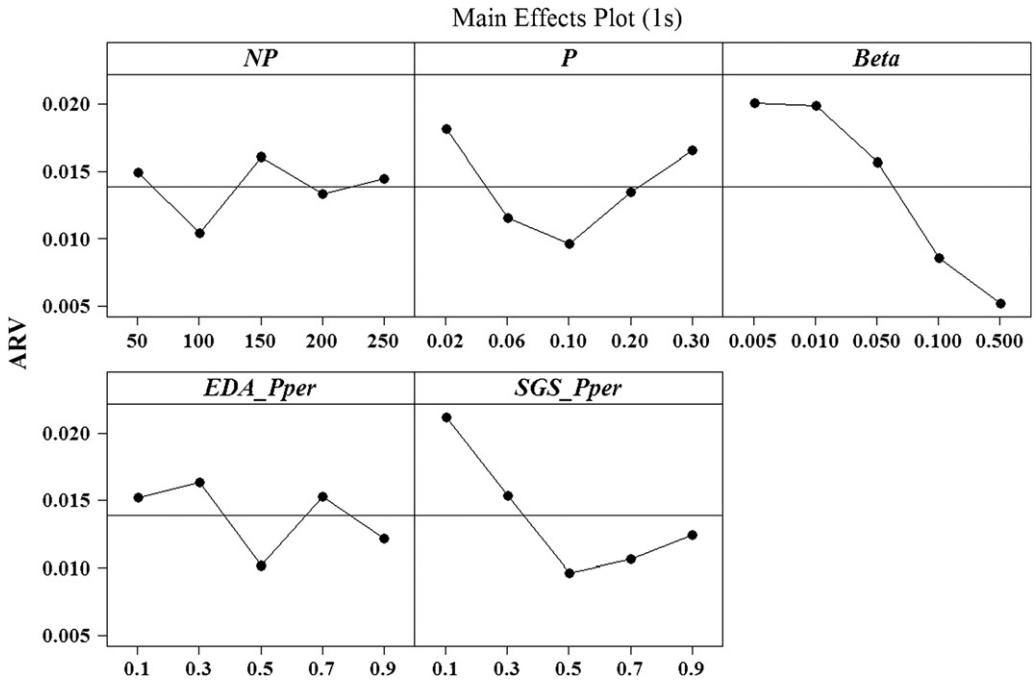


Fig. 11. Factor level trend with 1 s.

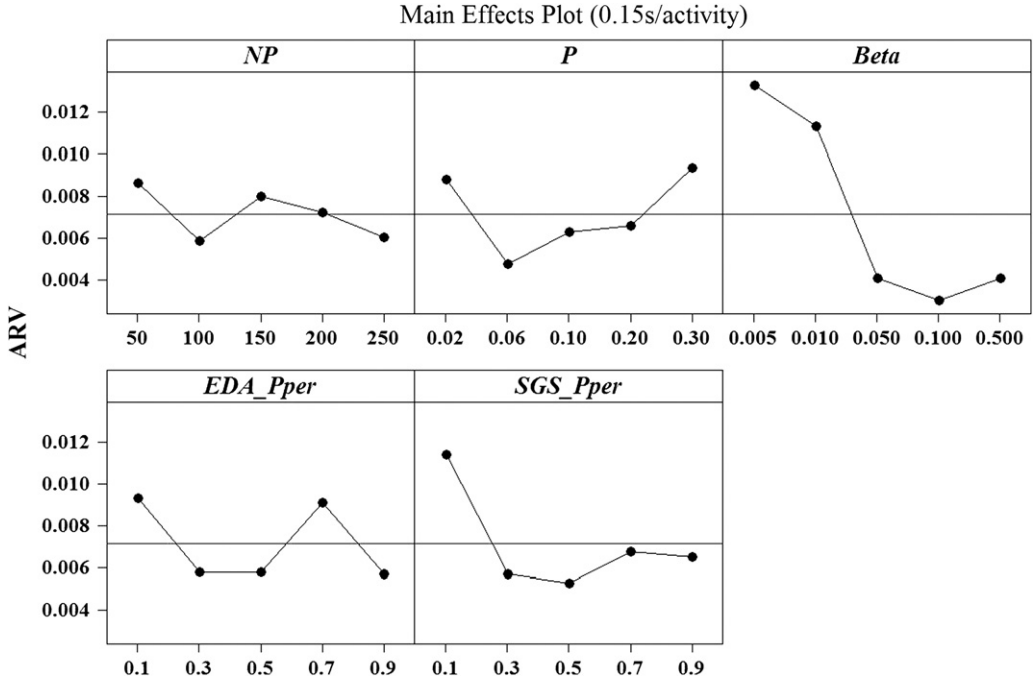


Fig. 12. Factor level trend with 0.15 s/activity.

Table 4
Response table for ARV with 5000 schedules.

Level	*NP*	*P*	*β*	*EDA_Pper*	*SGS_Pper*
1	0.03193	0.03538	0.03301	0.02903	0.04651
2	0.02746	0.02769	0.03303	0.03137	0.02927
3	0.03174	0.03032	0.03274	0.02810	0.02712
4	0.02798	0.02717	0.03153	0.03170	0.02371
5	0.03126	0.02981	0.02006	0.03019	0.02377
Delta	0.00447	0.00821	0.01296	0.00360	0.02280
Rank	4	3	2	5	1

Table 5
Response Table for ARV with 1 s.

Level	*NP*	*P*	*β*	*EDA_Pper*	*SGS_Pper*
1	0.014977	0.018164	0.020082	0.015252	0.021241
2	0.010442	0.011555	0.019861	0.016406	0.015351
3	0.016120	0.009591	0.015681	0.010199	0.009665
4	0.013351	0.013489	0.008554	0.015336	0.010639
5	0.014465	0.016555	0.005176	0.012162	0.012459
Delta	0.005678	0.008573	0.014906	0.006206	0.011576
Rank	5	3	1	4	2

$$\beta = -4.399e^{-6}z + 0.5821 \quad (22)$$

$$EDA\_Pper = 4.307e^{-6}z + 0.4253 \quad (23)$$

$$SGS\_Pper = -1.647e^{-6}z + 0.6462 \quad (24)$$

where $5000 \le z \le 104{,}895$.

5.2. Comparisons with existing algorithms

In this subsection, we compare the EDA with some existing algorithms based on the PSPLIB data sets to further show the effectiveness of the EDA.

We use Av. dev. to denote the average deviation from the lower bound. For the lower bound, the theoretically optimal values are used for set J10-20 and the critical-path based lower bounds are employed for set J30.

In Table 8, we compare the EDA with the simulated annealing developed by Józefowska et al. [4], the genetic algorithm developed by Alcaraz et al. [6], the hybrid scatter search developed by Ranjbar et al. [13], the hybrid rank-based evolutionary algorithm developed by Elloumi and Fortemps [12], the artificial immune system (AIS) developed by Van Peteghem and Vanhoucke [9], the two-phase genetic local search algorithm developed by Tseng and Chen [14], and the efficient hybrid genetic algorithm developed by Lova et al. [15], the GA developed by Van Peteghem and Vanhoucke [16]. We denote these algorithms as JSA, AGA, RSS, EFEA, VPVAIS, TCGLS, LHGA, and VPVGA, respectively. The stopping condition for the comparison is 5000 generated schedules. It can be seen from the results in Table 8 that VPVGA and VPVAIS performs better than our algorithm, and VPVAIS has better performance than our EDA except J12. However, the EDA is among the most competitive algorithms. It can be seen that the EDA outperforms all the other algorithms for all sets.

In Table 9, we compare the EDA with EFEA and another efficient simulated annealing developed by Bouleimen and Lecocq [5] denoted as BLSA. The stopping condition for the comparison is

Table 6
Response table for ARV with 0.15 s/acitivty.

Level	*NP*	*P*	β	*EDA_Pper*	*SGS_Pper*
1	0.008641	0.008797	0.013299	0.009357	0.011413
2	0.005864	0.004773	0.011323	0.005789	0.005738
3	0.008001	0.006270	0.004060	0.005794	0.005316
4	0.007248	0.006581	0.003024	0.009107	0.006779
5	0.006018	0.009350	0.004065	0.005724	0.006526
Delta	0.002777	0.004576	0.010275	0.003633	0.006097
Rank	5	3	1	4	2

Table 7
The best combination of parameters for the EDA.

	NP	*P*	β	*EDA_Pper*	*SGS_Pper*
5000 schedules	100	20%*NP*	0.5	0.5	0.7
1 s	100	10%*NP*	0.5	0.5	0.5
0.15 s/activity	100	6%*NP*	0.1	0.9	0.5

Table 8
Comparison with JSA, AGA, RSS, EFEA, VPVAIS, TCGLS, LHGA, and VPVGA (5000 generated schedules).

Av. dev (%)	J10	J12	J14	J16	J18	J20
EDA	0.12	0.14	0.43	0.59	0.90	1.28
VPVGA	**0.01**	0.09	0.22	**0.32**	**0.42**	**0.57**
VPVAIS	0.02	**0.07**	**0.20**	0.39	0.52	0.70
LHGA	0.06	0.17	0.32	0.44	0.63	0.87
EFEA	0.14	0.24	0.77	0.91	1.30	1.62
RSS	0.18	0.65	0.89	0.95	1.21	1.64
AGA	0.24	0.73	1.00	1.12	1.43	1.91
TCGLS	0.33	0.52	0.93	1.08	1.32	1.69
JSA	1.16	1.73	2.6	4.07	5.52	6.74

Table 9
Comparison with EFEA and BLSA (1 s CPU time).

J	Av. dev. (%)			Optimal rate (%)		
	EDA[a]	EFEA[b]	BLSA[c]	EDA[a]	EFEA[b]	BLSA[c]
10	**0.012**	0.08	0.21	**99.8**	98.5	96.3
12	**0.026**	0.12	0.19	**99.4**	97.5	91.2
14	**0.034**	0.42	0.92	**98.9**	90.6	82.6
16	**0.17**	0.60	1.43	**95.6**	85.9	72.8
18	**0.19**	0.71	1.85	**94.7**	83.1	69.4
20	**0.32**	1.02	2.10	**90.6**	76.7	66.9

[a] T7500 2.2 GHz, time limit 1 s.
[b] Pentium 3.00 GHz, time limit 1 s.
[c] Pentium 100 MHz, time limit five times the instance size (in seconds).

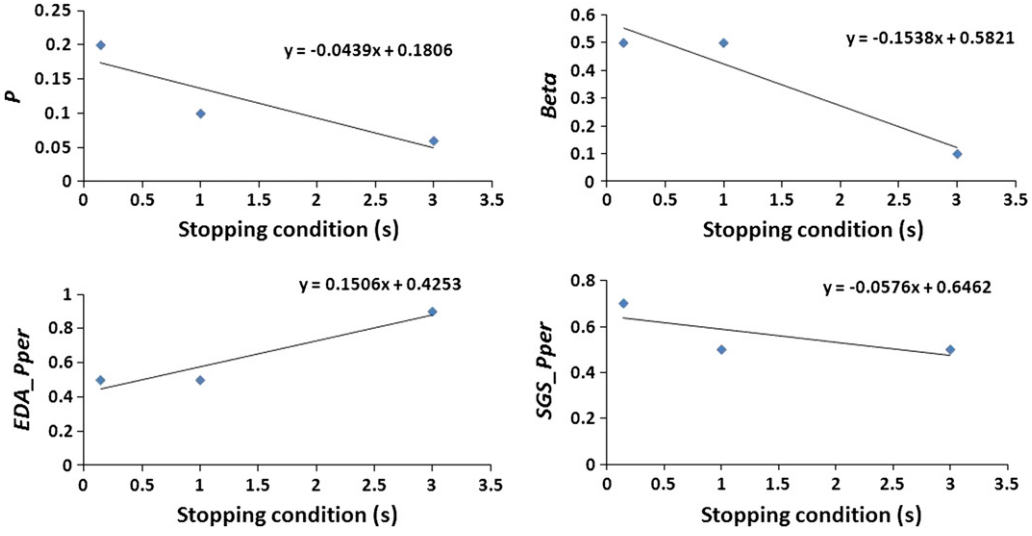


Fig. 13. Parameters and stopping conditions (the time of 5000 schedules equals to 0.143 s).

Table 10
Comparison with HGA and SDBB (1 s CPU time).

J	Av. dev. (%)			Feasible rate (%)			Optimal rate (%)		
	EDA	HGA	SDBB	EDA	HGA	SDBB	EDA	HGA	SDBB
10	0.012	0.06	**0.00**	100	100	100	99.8	98.7	**100**
12	**0.026**	0.14	0.12	100	100	100	**99.4**	97.3	98.2
14	**0.034**	0.44	1.46	100	100	99.6	**98.9**	89.8	85.7
16	**0.17**	0.59	3.81	100	100	99.5	**95.6**	87.8	69.5
18	**0.19**	0.99	7.48	100	100	98.0	**94.7**	78.3	57.4
20	**0.32**	1.21	11.51	100	100	96.4	**90.6**	73.3	47.3
30	**12.90**	16.93	57.22	86.1	86.3	55.8	n/a	n/a	n/a

Table 11
Comparison with EFEA, JPSO, and DDE (0.15 s/activity).

J	Av. dev. (%)				Optimal rate (%)			
	EDA	EFEA	JPSO	DDE	EDA	EFEA	JPSO	DDE
10	**0.005**	0.03	0.03	0.09	**99.8**	99.25	99.25	99.3
12	**0.016**	0.05	0.09	0.11	**99.6**	98.72	98.47	99.3
14	**0.044**	0.26	0.36	0.34	**98.3**	93.93	91.11	97.6
16	**0.075**	0.25	0.44	0.42	**96.9**	93.09	85.91	96.38
18	**0.16**	0.45	0.89	0.59	**95.4**	88.22	79.89	94.43
20	**0.26**	0.58	1.10	0.70	**92.2**	84.81	74.19	91.75

1 s CPU time. It can be seen from Table 9 that our EDA not only has a higher optimal rate than EFEA and BLSA but also has lower average deviation with less computation resource.

In Table 10, we compare the EDA with the genetic algorithm developed by Hartmann [3] and the truncated branch and bound developed by Sprecher and Drexl [2], which are denoted as HGA and SDBB, respectively. The stopping condition for the comparison is 1 s CPU time. As a branch and bound algorithm, SDBB can solve J10 optimally. However, as the scale of problem increases, the performance of SDBB drops rapidly. It can be seen from Table 10 that the EDA outperforms SDBB in all sets except J10, and our EDA outperform HGA in all tested sets except that the feasible rate of our EDA for J30 is slightly lower than that of HGA.

In Table 11, we compare the EDA with the particle swarm optimization developed by Jarboui et al. [8] denoted as JPSO, the Differential Evolution developed by Damak et al. [11] denoted as DDE, and EFEA. The stopping condition for the comparison is 0.15 s CPU time per activity. It can be seen from Table 11 that our EDA not only has lower average deviation but also has better optimal rate than EFEA, JPSO and DDE for all the problem sets.

As an incremental learning method the EDA needs effort to find and track the most promising area at the beginning of the search procedure. As a result, some algorithms outperform the EDA when small computational effort (i.e. 5000 schedules as stopping condition) is allowed. However, the EDA can provide very competitive results when the allowed computational effort becomes relatively larger (1 s CPU time or 0.15 s/activity as stopping condition). All in all, the above comparisons between the EDA and many existing algorithms show that the proposed EDA is an effective algorithm for solving the MRCPSP.

6. Conclusion

This was the first reported work to design an estimation of distribution algorithm for solving the multi-mode resource-constrained project scheduling problem. By using an encoding scheme based on the activity-mode list and a decoding scheme based on the multi-mode serial SGS, the EDA could be applied to the MRCPSP conveniently. By adopting a novel probability model and updating mechanism, the promising area could be tracked effectively to keep finding better solution. By applying the combined local search with multi-mode permutation based local search and multi-mode forward–backward improvement, the exploitation could be enhanced as well. Based on the DOE method, suitable parameter settings were determined, and the guidelines to set parameters of the EDA with different stopping conditions were provided. Simulation results based on the PSPLIB benchmarks and comparisons with some existing algorithms demonstrated the effectiveness of the proposed EDA. The further work is to develop an adaptive EDA with parameter learning mechanism and to extend the EDA method to solve other scheduling problems.

Acknowledgments

The authors sincerely thank the Editor-in-Chief and the reviewers for providing the constructive and helpful comments to improve the quality of this paper. The authors also would like to thank Prof. Rainer Kolisch of the Technical University of Munich, Germany, for providing useful information about PSPLIB. This research is partially supported by National Science Foundation of China (Grant no. 70871065, 60834004) and the Program for New Century Excellent Talents in University (NCET-10-0505) as well as the Doctoral Program Foundation of Institutions of Higher Education of China (20100002110014).

References

[1] Kolisch R, Drexl A. Local search for nonpreemptive multi-mode resource-constrained project scheduling. IIE Trans 1997;29(12):987–99.
[2] Sprecher A, Drexl A. Solving multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. Eur J Oper Res 1998;107(2):431–50.
[3] Hartmann S. Project scheduling with multiple modes: a genetic algorithm. Ann Oper Res 2001;102(1–4):111–35.
[4] Józefowska J, Mika M, Różycki R, Waligóra G, Weglarz J. Simulated annealing for multi-mode resource-constrained project scheduling. Ann Oper Res 2001;102(1–4):137–55.
[5] Bouleimen K, Lecocq H. A new efficient simulated annealing algorithm for resource-constrained project scheduling problem and its multiple mode version. Eur J Oper Res 2003;149(2):268–81.
[6] Alcaraz J, Maroto C, Ruiz R. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. J Oper Res Soc 2003;54(6):614–26.
[7] Zhang H, Tam CM, Li H. Mulitmode project scheduling based on particle swarm optimization. Comput-Aided Civil Infrastruct Eng 2006;21(2):93–103.
[8] Jarboui B, Damak N, Siarry P, Rebai A. A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. Appl Math Comput 2008;195(1):299–308.
[9] Van Peteghem V, Vanhoucke M. An artificial immune system for the multi-mode resource-constrained project scheduling problem. In: Proceedings of the 9th European conference, EvoCOP 2009. Berlin: Springer Press; 2009. p. 85–96.
[10] Wauters T, Verbeeck K, Berghe GV, De Causmaecker P. A multi-agent learning approach for the multi-mode resource-constrained project scheduling problem. In: Proceedings of the of 8th international conference on autonomous agents and multiagent systems; 2009.
[11] Damak N, Jarboui B, Siarry P, Loukil T. Differential evolution for solving multi-mode resource-constrained project scheduling problems. Comput Oper Res 2009;36(10):2653–9.
[12] Elloumi S, Fortemps P. A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. Eur J Oper Res 2010;205(1):31–41.
[13] Ranjbar M, Reyck B, De, Kianfar F. A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. Eur J Oper Res 2009;193(1):35–48.
[14] Tseng LY, Chen SC. Two-phase genetic local search algorithm for the multi-mode resource-constrained project scheduling problem. IEEE Trans Evol Comput 2009;13(4):848–57.
[15] Lova A, Tormos P, Cervantes M, Barber F. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. Int J Prod Econ 2009;117(2):302–16.

[16] Van Peteghem V, Vanhoucke M. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problems. Eur J Oper Res 2010;201(2):409–18.
[17] Larraanaga P, Lozano JA. Estimation of distribution algorithms, a new tool for evolutionary computation. Boston: Kluwer Press; 2002.
[18] Saeys Y, Degroeve S, Aeyels D, Van de Peer Y, Rouze P. Fast feature selection using a simple estimation of distribution algorithm: a case study on splice site prediction. Bioinformatics 2003;19(2):179–88.
[19] Jarboui B, Eddaly M, Siarry P. An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. Comput Oper Res 2009;36(10):2638–46.
[20] Aickelin U, Burke EK, Li J. An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. J Oper Res Soc 2007;58(13):1574–85.
[21] Zhang Q, Sun J, Tsang E, Ford J. Combination of guided local search and estimation of distribution algorithm for solving quadratic assignment problem. In: Proceedings of the bird of a feather workshops, genetic and evolutionary computation conference; 2003. p. 42–8.
[22] Simionescu PA, Beale DG, Dozier GV. Teeth-number synthesis of a multispeed planetary transmission using an estimation of distribution algorithm. J Mech Des 2006;128(1):108–15.
[23] Cesar Jr. RM, Bengoetxea E, Bloch I, Larranaga P. Inexact graph matching for model-based recognition: Evaluation and comparison of optimization algorithms. Pattern Recognition 2005;38(12):2099–113.
[24] Sagarna R, Lozano J. On the performance of estimation of distribution algorithms applied to software testing. Appl Artif Intell 2005;19(5):457–89.
[25] Sprecher A, Hartmann S, Drexl A. An exact algorithm for project scheduling with multiple modes. OR Spektrum 1997;19(3):195–203.
[26] Kolisch R, Hartmann S. Heuristic algorithms for solving the resource-constrained project scheduling problem: classification, computational, analysis. In: Weglarz J, editor. Project scheduling: recent models, algorithms and applications. Berlin: Kluwer Press; 1999. p. 147–78.
[27] Kolisch R. Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. Eur J Oper Res 1996;90(2):320–33.
[28] Lozano JA, Larranaga P, Inza I, Bengoetxea E. Towards a new evolutionary computation advances on estimation of distribution algorithms. Berlin: Springer Press; 2006.
[29] Li KY, Willis RJ. An iterative scheduling technique for resource constrained project scheduling. Eur J Oper Res 1992;56(3):370–9.
[30] Kolisch R, Sprecher A. PSPLIB-a project scheduling problem library. Eur J Oper Res 1997;96(1):205–16.
[31] Montgomery DC. Design and analysis of experiments. Arizona: John Wiley & Sons Press; 2005.