Expert Systems with Applications 39 (2012) 2451–2460

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

A hybrid estimation of distribution algorithm for solving the 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

Keywords:
Resource-constrained project scheduling
Estimation of distribution algorithm
Probability model
Permutation based local search
Hybrid algorithm

ABSTRACT

In this paper, a hybrid estimation of distribution algorithm (HEDA) is proposed to solve the resource-constrained project scheduling problem (RCPSP). In the HEDA, the individuals are encoded based on the extended active list (EAL) and decoded by serial schedule generation scheme (SGS), and a novel probability model updating mechanism is proposed for well sampling the promising searching region. To further improve the searching quality, a Forward–Backward iteration (FBI) and a permutation based local search method (PBLS) are incorporated into the EDA based search to enhance the exploitation ability. Simulation results based on benchmarks and comparisons with some existing algorithms demonstrate the effectiveness of the proposed HEDA.

1. Introduction

The resource-constrained project scheduling problem (RCPSP) is concerned with single-item or small batch production where scarce resources have to be allocated to dependent activities over time (Brucker, Drexl, Mohring, Neumann & Pesch, 1999). A project is composed of activities which need to be finished in their durations. And there are precedence relationships between activities. For a resource-constrained project, there are some kinds of renewable resources which have a constant capacity for every period of project. Given the total units of renewable resource in each period of project, the RCPSP is to study the reasonable utilization of resource and the scheduling of project activities to minimize the makespan or total flow time over all activities of the project. As an active area, research on the RCPSP has been extended to a variety of academic and engineering fields, such as medical research (Hartmann, 1997), software development (Alba & Francisco Chicano, 2007), audit scheduling (Dodin, Elimam, & Rolland, 1998), market research interviewers scheduling (Hartmann, 1999), and so on. As for the computational complexity, Blazewicz, Lenstra, and Rinnooy (1983) proved that it was NP-hard. As a result, the conventional exact algorithms (e.g. branch and bound method) can only solve small-size instances of the RCPSP optimally in acceptable time. For the large-size instances, heuristics are more widely used to obtain optimal or sub-optimal solutions with acceptable time and memory requirements.

In general, heuristic methods can be divided into two groups: constructive methods and meta-heuristic methods. In the first group, priority rule based heuristics are most popular, which employ priority rules to decide which activity to be selected from the decision set in each step of constructing a solution. After that, schedule generation scheme (SGS) is adopted to construct one or more schedules based on the constructed solution. The SGS is the core of most heuristic procedures for the RCPSP, which is used to decode representation of a solution to a schedule. There are two kinds of SGS frequently used in literature: serial SGS based on activity incrementation, and parallel SGS based on time incrementation (Kolisch, 1996b). So far, many priority rules have been proposed, e.g., greatest rank positional weight (GRPW) rule, most total successors (MTS) rule, shortest processing time (SPT) rule (Alvarez-Valdes & Tamarit, 1989), minimum slack (MSLK) rule, latest start time (LFT) rule (Davis & Patterson, 1975), latest start time (LST) rule (Kolisch, 1995), worst case slack (WCS) rule (Kolisch, 1996a), resource scheduling method (RSM) rule (Shaffer, Ritter, & Meyer, 1965). In addition, Cooper (1977) studied different single pass methods in which only one SGS and one priority rule were employed, and some conclusions about different priority rules were provided. Later, multi-priority rule methods that employed SGS several times (Thomas & Salhi, 1997) and sampling methods that adopted different priority rules in different time (Drexl, 1991) were proposed.

With the development of computer technology and computational intelligence, meta-heuristics, such as genetic algorithm (GA), simulated annealing (SA), tabu search (TS), particle swarm optimization (PSO), scatter search (SS), have been developed for solving the RCPSP. In GA, Hartmann (1998) proposed a permutation based GA, which adopted regret-based sampling method and priority rule to produce initial population; Alcaraz and Maroto (2001) presented an activity list based GA in which a gene was

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

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.08.095

added to decide Forward or Backward schedule generation scheme (SGS) to be used. Later, Hartmann (2002) proposed an adaptive GA, in which a gene was adopted to decide parallel SGS or serial SGS to be used. In SA, Boctor (1996) proposed an active list based SA to solve the RCPSP, where serial SGS was used to generate schedule and insert operation was employed as local search; Cho and Kim (1997) introduced a random key based SA, where some activities were delayed on purpose to expand search space; Bouleimen and Lecocq (2003) proposed a global shift operation-based SA, which adopted multiple cooling chains with different initial solution. In TS, Klein (2000) proposed a Forward–Backward tabu search to solve the time-varying RCPSP, where active list and serial SGS were adopted; Artigues, Michelon, and Reusser (2003) introduced an insert based TS, where the abandoned solutions were inserted based on a flow network model; Nonobe and Ibaraki (2001) adopted TS to solve the RCPSP, where shift moves and a specific neighborhood reduction mechanism were developed to improve the standard TS. In PSO, Zhang, Li, Li, and Huang (2005) proposed both permutation-based particle swarm optimization and priority-based PSO for solving the RCPSP. In SS, Debels, De Reyck, Leus, and Vanhoucke (2006) developed a hybrid SS to solve the RCPSP, in which electromagnetism heuristic and Forward/Backward shift of individual activities were adopted to enhance the SS.

Estimation of distribution algorithm (EDA) is a newly proposed stochastic optimization algorithm (Larranaga & Lozano, 2002). Unlike GA which explicitly applies genetic operators (e.g. crossover, mutation) to produce new generation, EDA reproduces new population implicitly. In EDA, statistical information based on the searching experience is picked up to build a probability model of the most promising area, and new generation is produced by sampling the probability model. In each generation, good individuals in the new generation are selected to update the probability model. Please refer Larranaga and Lozano (2002) for more details about EDA. So far, EDA has been applied to a variety of academic and engineering optimization problems, such as feature selection (Saeys, Degroeve, Aeyels, Van de Peer, & Rouze, 2003), flow-shop scheduling (Jarboui, Eddaly, & Siarry, 2009), nurse rostering (Aickelin, Burke, & Li, 2007), quadratic assignment problem (Zhang, Sun, Tsang, & Ford, 2003), multispeed planetary transmission design (Simionescu, Beale, & Dozier, 2006), inexact graph matching (Cesar, Bengoetxea, Bloch, & Larranaga, 2005), and software testing (Sagarna & Lozano, 2005). In this paper, we will propose a hybrid EDA (HEDA) for solving the RCPSP with a criterion to minimize the makespan. In particular, a new encoding scheme based on an extended active list will be proposed, and a novel probability model updating mechanism will be developed in the procedure of HEDA to help identify the most promising area, and a Forward–Backward iteration (FBI) and a permutation based local search method (PBLS) will be applied to the best individuals to exploit the neighborhood of the best individuals. Computational results and comparisons will demonstrate the effectiveness of the HEDA.

The remainder of the paper is organized as follows: In Section 2, the RCPSP is described. In Section 3, the basic EDA is introduced. Then, the hybrid EDA for the RCPSP 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. Resource-constrained project scheduling problem

The RCPSP is to study the reasonable utilization of resource and the scheduling of project activities to optimize the makespan. Generally, the RCPSP can be stated as follows. A project consists of J activities labeled as $j = 1, \ldots, J$, where the duration of activity j is denoted by d_j. There are precedence relationships between some activities of project. The precedence relationship is given by sets

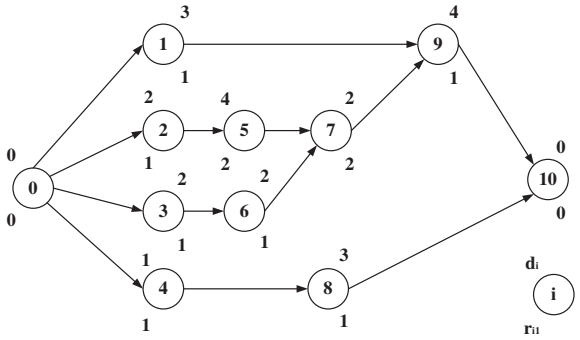

Fig. 1. An example of the RCPSP.

of immediate predecessors P_j indicating that an activity j may not be started before each of its predecessors $i \in P_j$ is completed. The set of renewable resources is referred as K^ρ. For each resource $k \in K^\rho$, the per-period-availability is assumed to be constant R_k^ρ. Activity j requires r_{jk} units of resource k in each period of its non-preemptable duration. The activities $j = 0$ and $j = J + 1$ are dummy activities, which represent the start and end of the project respectively. It assumes that the dummy activities do not request any resource and their durations are equal to zero. The set of all activities including the dummy activities is denoted as $J^+ = \{0, \ldots, J + 1\}$. The objective is to minimize the makespan of the project.

In Fig. 1, a simple example with 11 activities (including two dummy activities $j = 0$ and $j = J + 1$) is illustrated. In this example, only one kind of renewable resource is used, and the number of available renewable resources in each period is two. In Fig. 2, a schedule of this project is shown. Since both precedence constraint and resource constraint are satisfied, it is a feasible schedule with makespan 18.

3. Estimation of distribution algorithm

During the last decade, estimation of distribution algorithm (EDA) has emerged as a general framework for a set of probability distribution based optimization algorithms (Larranaga and Lozano, 2002). With the tool of statistical analysis, EDA tries to predict the movement of population in the searching space and to estimate the underlying probability distribution of encoded variables of the elite individuals.

The general framework of the EDA is illustrated in Fig. 3. After generating of the initial population and initializing of the probability matrix, an iterative procedure is carried out to estimate the distribution of the optimal solution until the stopping condition is met.

The core of the EDA procedure is to estimate the probability distribution. Due to the difference of problem types, different probability models can be chosen to estimate the underlying probability distribution. Based on the searching mechanism of EDA, we will propose a special probability model and an updating mechanism in this paper to solve the RCPSP.

4. Hybrid EDA (HEDA) for RCPSP

First, we introduce the encoding scheme, the probability model, the probability generating mechanism, local search strategy, and the learning-based updating mechanism. Then, we present the framework of proposed HEDA.

4.1. Encoding scheme and schedule generation scheme

The HEDA does not directly operate on a schedule but on the representation of a schedule. Ideally, an individual representation

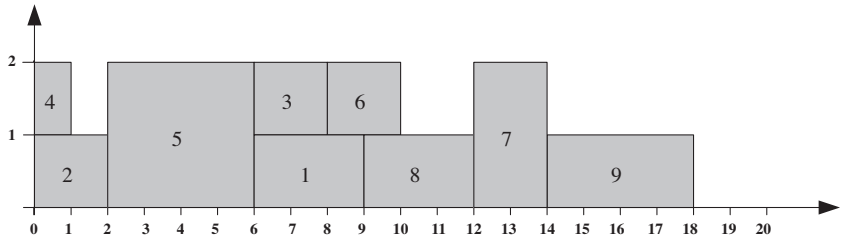

Fig. 2. Illustration of a feasible schedule.

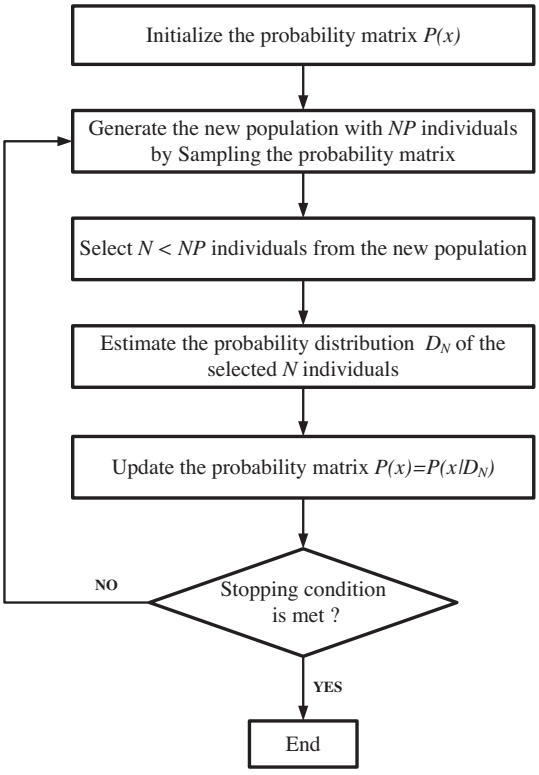

Fig. 3. The general framework of the EDA.

should meet the following requirements (Palmer & Kershenbaum, 1994): (1) The transformation between solutions should be computationally fast; (2) For each solution in the original space, there is a solution in the encoded space; (3) Each encoded solution corresponds to one feasible solution in the original space; (4) All solutions in the original space should be represented by the same number of encoded solutions; (5) Small changes in the encoded solution should result in small changes in the solution itself.

Kolisch and Hartmann (1999) concluded from experimental tests that procedures based on activity list representations outperformed the other procedures. Inspired by their work, we propose an extended activity list (EAL) in this paper. The EAL contains three parts: an activity list (AL) $\Pi : [\pi_1, \pi_2, \cdots, \pi_J]$, a start time list of every activity $ST : [st_1, st_2, \cdots, st_J]$, and a finish time list of every activity $FT : [ft_1, ft_2, \cdots, ft_J]$. In an EAL, the start (finish) time of activity π_i is st_i (ft_i). Considering the RCPSP example illustrated in Fig. 1, a feasible schedule and its EAL representation are depicted in Fig. 4.

With the above encoding scheme, a schedule generation scheme (SGS) should be used to transform the EAL to a schedule. After a new EAL is generated, SGS is used to evaluate the EAL. Once the EAL is changed (e.g. performing local search) the three parts of EAL could be synchronized by SGS without any extra computation.

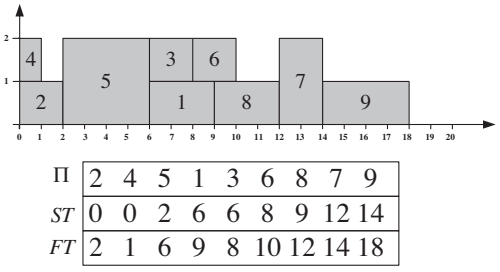

Fig. 4. A schedule and its EAL.

With the help of SGS, the EAL can satisfy the five requirements mentioned above easily. There are two kinds of SGS in the literature, including serial SGS and parallel SGS. Since parallel SGS is sometimes unable to obtain an optimal schedule (Kolisch, 1996b), the serial SGS is adopted in our procedure.

Comparing with AL, the EAL needs twice more additional memory to store the *ST* and *FT*. However, the *ST* (*FT*) will accelerate the Forward (Backward) scheduling process in the Forward–Backward iteration. In a single iteration, AL should be scheduled three times. With the help of *ST* and *FT*, EAL only need to be scheduled twice (see section 4.5). This character can speed up computational process of the algorithm, which makes the proposed HEDA more convenient and effective to solve the RCPSP.

4.2. Probability model

Different from GA that produces offspring through crossover and mutation operators, EDA does it by sampling according to a probability model which has a great effect on the performances of EDA. How to construct the probability model is the key issue to design EDA (Lozano, Larra aga, & Bengoetxea, 2006).

In this paper, the probability model is designed as the following $J \times J$ probability matrix.

$$Prob(t) = \begin{pmatrix} prob_{11} & \cdots & prob_{1J} \\ \vdots & \ddots & \vdots \\ prob_{J1} & \cdots & prob_{JJ} \end{pmatrix} \quad (1)$$

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

The probability matrix is initialized as follows, which ensures that the whole solution space can be sampled uniformly.

$$Prob(0) = \begin{pmatrix} \frac{1}{J} & \cdots & \frac{1}{J} \\ \vdots & \ddots & \vdots \\ \frac{1}{J} & \cdots & \frac{1}{J} \end{pmatrix} \quad (2)$$

4.3. Probability generating mechanism

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

At every position i, the selection probability of activity j i.e. p_{ji} is calculated according to probability matrix $Prob^t$ over the set of eligible activities D, that is

$$p_{ji} = \frac{prob_{ji}}{\sum_{h=1}^{D} prob_{hi}} \tag{3}$$

If activity j has already been placed in some positions, the whole line $prob_{j1}, prob_{j2}, \ldots, prob_{jJ}$ of probabilistic matrix $Prob$ will be set as zero.

4.4. Local search strategy

To enhance the exploitation ability, a permutation-based local search strategy PBLS controlled by a threshold $Pper$ is proposed to explore the neighborhood of an individual. This operator is a variation of SWAP operator, but the PBLS does not break the precedence feasibility of the EAL if it is precedence feasible before performing local search. The procedure of PBLS is described in Fig. 5.

4.5. Forward–Backward iteration and the speed-up evaluation of EAL

Forward–Backward iteration (FBI) is an effective technique for the RCPSP (Li & Willis, 1992). Basically, the procedure iteratively employs SGS to Forward and Backward schedule until there is no further improvement to the makespan of the project. The activity finish times of a Forward schedule determine the activity priorities for the next Backward schedule. Similarly, the activity start times of a Backward schedule determine the activity priorities for the Forward schedule. The FBI can be easily incorporated into many algorithms for the RCPSP to improve the solution quality. In Fig. 6(a), a single iteration step is used to illustrate procedure of the FBI.

```
Procedure PBLS
For (i=1,2,…,n)
{
    Randomly generate q where 0<q<1;
    If (q<Pper)
    {
        If ( π_i is not the predecessor of π_{i+1} )
        {
            Swap π_i and π_{i+1};
            Evaluate the new EAL;
            If (Makespan is improved)
            {
                Record current EAL and makespan;
            }
        }
    }
}
```

Fig. 5. Procedure of the PBLS.

To reduce the makespan of Fig. 2, the FBI is used by shifting each activity to the right as much as possible in decreasing order of activity end times. For example, Activity 9 and Activity 7 cannot be scheduled later. Activity 8 can be right shifted to start at time 15. Activity 6 can be shifted two time units and start at time 10. Since the right shift of Activity 8 has made some additional resources available, Activity 1 can be shifted three time units to start at time 9. Activity 2, Activity 3 and Activity 5 can be shifted two time units. Finally, Activity 4 is shifted to time 14. In this way, we obtain a schedule with a makespan of 16 units. Further improvements of the schedule are possible by shifting activities as much as possible to the left. This process reduces the makespan by one further time unit, as illustrated at the bottom of Fig. 6(a).

If the EAL is adopted, two sort operations and two schedule operations are needed in a single iteration of FBI as shown in Fig. 6(b). However, if only AL is adopted, two calculating operations, two sort operations, and three schedule operations are needed in a single iteration of FBI as shown in Fig. 6(c).

Since the time complexity of quick sort is $O(n \log n)$, the time complexity of serial SGS is $O(n^2)$, and the time complexity of computing FT (ST) according to ST (FT) is $O(n)$, the computational time of the FBI with EAL will be about two third of the computational time of the FBI with AL.

4.6. Updating mechanism

In this paper, a population based updating mechanism is proposed to update the probabilistic matrix $Prob$. First, a number of NP individuals are generated according to the matrix $Prob$, and FBI and PBLS are employed to renew the best P individual. Then, the best P individuals are chosen to update the probabilistic matrix $Prob$ according to the following equation:

$$prob_{ji}(t+1) = (1-\beta)\cdot prob_{ji}(t) + \frac{\beta}{P}\sum_{k=1}^{P} I_{ji}^{k}, \quad (1 \le i,j \le J; 0 < \beta < 1) \tag{4}$$

where β is the learning speed and I_{ji}^{k} is the indicator function of the kth best individual.

By taking problem specific knowledge about the precedence constraint in the RCPSP into account, we design a novel updating mechanism (NUM) of probability model. That is, the probabilistic matrix $Prob$ is updated according to Eq. (4), and the indicator function of the kth best individual is calculated as follows:

$$I_{ji}^{k} = \begin{cases} \lambda^{|i_0 - i|} & \text{if activity is placed at position} i_0 \; and \; |i_0 - i| \leqslant R \\ 0 & else \end{cases} \tag{5}$$

where R is the affecting radius, in which activities affect the selection probability of activity j in position i; and λ is the descending rate satisfying $0 < \lambda < 1$.

After updating, the probabilistic matrix $Prob$ will be normalized as follows:

$$prob_{ji} = \frac{prob_{ji}}{\sum_{h=1}^{J} prob_{hi}} \tag{6}$$

Definition 1. If the activities of an AL can be scheduled one by one according to the AL sequence, we call the AL is precedence feasible.

Definition 2. If the AL part of an EAL is precedence feasible, we call the EAL is precedence feasible.

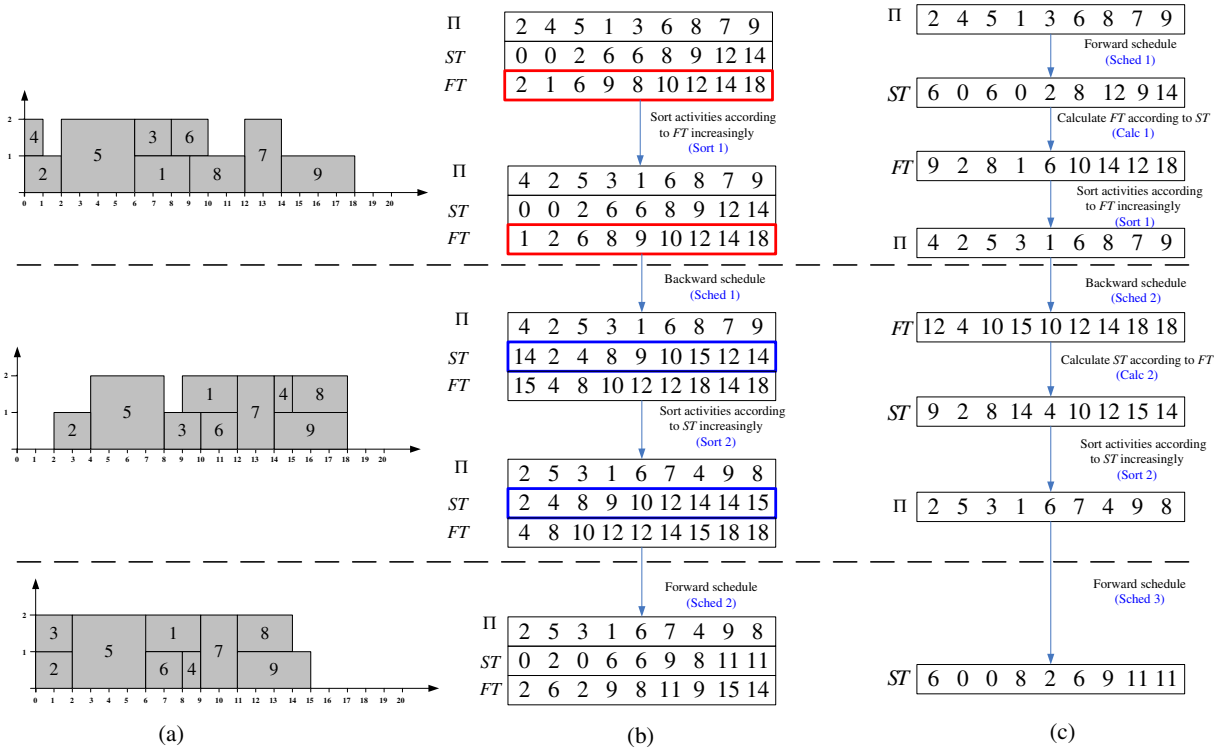

Fig. 6. FBI and the speed-up effect of EAL.

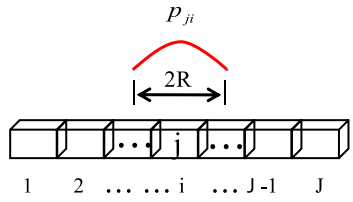

Fig. 7. The dependence of the selection probability of activity j in the position i.

The RCPSP is precedence constrained, that is, each activity in RCPSP should not be started before the time all its predecessors have been finished and each activity must be finished before the time all its successors can be start. Clearly, in a precedence EAL, the position i of an activity j should be constrained by the number of its predecessors (both direct and indirect) AP_j and number of its successors (both direct and indirect) AS_j, that is:

$$AP_j < i < AS_j \tag{7}$$

So, activity j can be placed in any positions between AP_j and AS_j without violating precedence relationships. As a result, it may be good to place activity j in the positions near i, if it is good to place activity j in the position i. In this mechanism, the selection probability of activity j in the position i, i.e. p_{ji}, not only depends on the element $prob_{ji}$ of probability matrix, but also depends on the elements $prob_{j(i-R)}, \ldots, prob_{j(i-1)}, prob_{j(i+1)}, \ldots, prob_{j(i+R)}$.

However, the dependence decreases exponentially as the distance $|i - i_0|$ increases. Assume the maximal affecting range (affecting radius) is calculated as $\lambda^R \leqslant 0.1$, which means that we omit dependence effect when it is less than or equal to 0.1. So, we can get $R \geqslant \frac{-1}{\lg\lambda}$. Accordingly, we set $R = \lceil \frac{-1}{\lg\lambda} \rceil$. The dependence is illustrated in Fig. 7.This updating mechanism utilizes the precedence relationship to help the algorithm sample the most promising area more preciously. With the help of this updating mechanism, the algorithm can also avoid premature convergence and avoid being trapped in some local minima.

4.7. Procedure of HEDA

With the above design, the procedure of the HEDA is summarized as follows: First, the probability matrix is initialized uniformly. Then the new population with NP individuals is generated by sampling the probability matrix using PGM. During every generation, all the individuals are evaluated by serial SGS and sorted in ascent order according to the makespan values. The best P ($P < NP$) individuals are selected from the population. Then the FBI is applied to the selected P individuals to improve the makespan value. Following the PBLS that is applied for further improvement, the indicator function I_{ji}^k of each individual k is calculated using NUM strategy. Finally, the probability matrix $Prob(t)$ is updated.

Straightforwardly, the framework of the HEDA is illustrated in Fig. 8.

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

5. Computational results and comparisons

We code the procedure in Visual C++ 2005 on IBM Thinkpad T61 with a Core 2 T7500 2.2 GHz processor, and use the well-known data sets PSPLIB for testing which are generated by the problem generator ProGen designed by (Kolisch and Sprecher, 1996). This data set contains subsets J30, J60, J90, and J120, which have 30, 60, 90, and 120 activities, respectively. The set J30, J60 and J90 consist of 480 instances, and J120 consists of 600 instances. All the problem sets are designed with different network complexity, resource factor, and resource strength. In this paper, we use J30, J60, and J120 to test the performance of HEDA and to compare with some existing algorithms in literature.

5.1. Parameters setting

The proposed HEDA contains five key parameters: the population size of each generation (NP), the number of selected individual

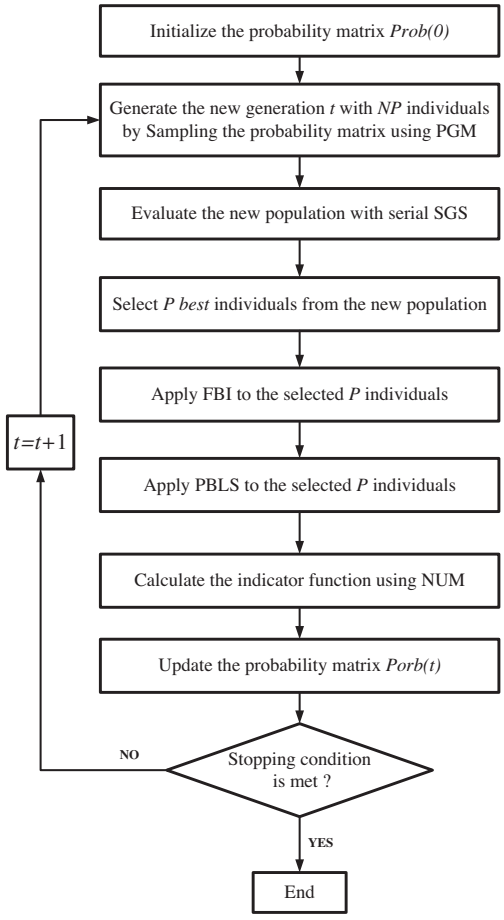

Fig. 8. The framework of the HEDA.

to update the probability matrix (P), the learning speed (β), the PBLS accept rate (*Pper*), and the descending rate (λ). First, we use Taguchi method of design of experiment (DOE) (Montgomery, 2005) to determine a set of suitable parameters for the HEDA. Combinations of different values of these parameters are shown in Table 1. We randomly choose 48 instances from J30, 48 instances from J60, and 60 instances from J120 to carry out the DOE test.

We run the HEDA 20 times independently for each instance and set the maximum number 5,000 of the schedules as the stopping criterion. The average response variable (ARV) values for J30, J60, and J120 are the following average deviation values for $N = 48$, 48, 60 instances, respectively.

$$ARV = \sum_{i=1}^{N}\sum_{j=1}^{20}\frac{(Makespan_{ij} - LB)}{LB}\Big/(20 \cdot N) \quad (8)$$

where $Makespan_{ij}$ is the makespan of the ith instance obtained by the HEDA in the jth run; LB is the lower bound to compare with.

Table 1
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
Pper	0	0.25	0.5	0.75	1
λ	0.3	0.5	0.7	0.8	0.9

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 treatment is 25, the number of parameters is 5, and the number of factor levels is 5. The orthogonal arrays for J30, J60, J120 are listed in Table 2.

According to the orthogonal table, we illustrate the trend of each factor level for J30, J60, and J120 in Figs. 9–11, respectively. Then, we figure out the response value changes of each parameter to analyze the significance rank of each parameter for J30, J60, and J120. The results are listed in Tables 3–5, respectively.

From the response tables, it can be seen that: P is the most significant parameter among the five parameters; the significant rank of *NP* is the 5th for J30 and J60, and 4th for J120, respectively; the significant rank of β is 2nd for J30 and J60, 3rd for J120, respectively; the significant rank of *Pper* is 4th for J30, 3rd for J60, and 1st for J120, respectively; the significant rank of r is 3rd for J30, 4th for J60, and 5th for J120, respectively.

According to the factor level trend, the best combinations of parameter values for J30, J60, and J120 are determined, which are listed in Table 6.

5.2. Computational results of HEDA

Next, we set the maximum number of schedules as 500, 1,000 and 5,000 respectively, and we run the HEDA 20 times independently for every instance with parameter setting listed in Table 6. The statistical results are summarized in Table 7, where Ave.LB.-Dev, Min.LB.Dev, and Max.LB.Dev are the average, minimum and maximum percentage deviations from the lower bounds, and Var is the variance of the deviations, and Ave.CPU, Min.CPU, and Max.CPU denote the average, minimum and maximum CPU times. As for the lower bound, the theoretically optimal values are used for set J30 and the critical-path based lower bounds reported by Stinson, Davis, and Khumawala (1978) are employed for set J60 and set J120.

From Table 7, it can be seen that the derivation values and CPU times increase as the size of the problem increases. It shows that the RCPSP with large scale is more difficult to solve since the prob-

Table 2
Orthogonal table for the HEDA.

Experiment Number	Factors					ARV		
	NP	*P*	β	*Pper*	λ	J30	J60	J120
1	1	1	1	1	1	0.001641	0.116161	0.344836
2	1	2	2	2	2	0.001453	0.114559	0.3393
3	1	3	3	3	3	0.001562	0.114875	0.340159
4	1	4	4	4	4	0.002108	0.115628	0.343237
5	1	5	5	5	5	0.002751	0.116573	0.345605
6	2	1	2	3	4	0.001488	0.11391	0.337202
7	2	2	3	4	5	0.001698	0.114429	0.338661
8	2	3	4	5	1	0.001681	0.114089	0.339454
9	2	4	5	1	2	0.001516	0.11581	0.347577
10	2	5	1	2	3	0.002387	0.116853	0.343872
11	3	1	3	5	2	0.001228	0.113396	0.336613
12	3	2	4	1	3	0.001442	0.115671	0.345594
13	3	3	5	2	4	0.001582	0.114663	0.340509
14	3	4	1	3	5	0.002313	0.116038	0.34221
15	3	5	2	4	1	0.002589	0.117001	0.341511
16	4	1	4	2	5	0.001633	0.114896	0.33846
17	4	2	5	3	1	0.001082	0.113712	0.337531
18	4	3	1	4	2	0.002035	0.115143	0.344321
19	4	4	2	5	3	0.002377	0.11616	0.345128
20	4	5	3	1	4	0.001795	0.115819	0.344133
21	5	1	5	4	3	0.001119	0.113007	0.336013
22	5	2	1	5	4	0.001949	0.114275	0.338518
23	5	3	2	1	5	0.001948	0.115664	0.345704
24	5	4	3	2	1	0.002053	0.115694	0.342337
25	5	5	4	3	2	0.002166	0.116377	0.340333

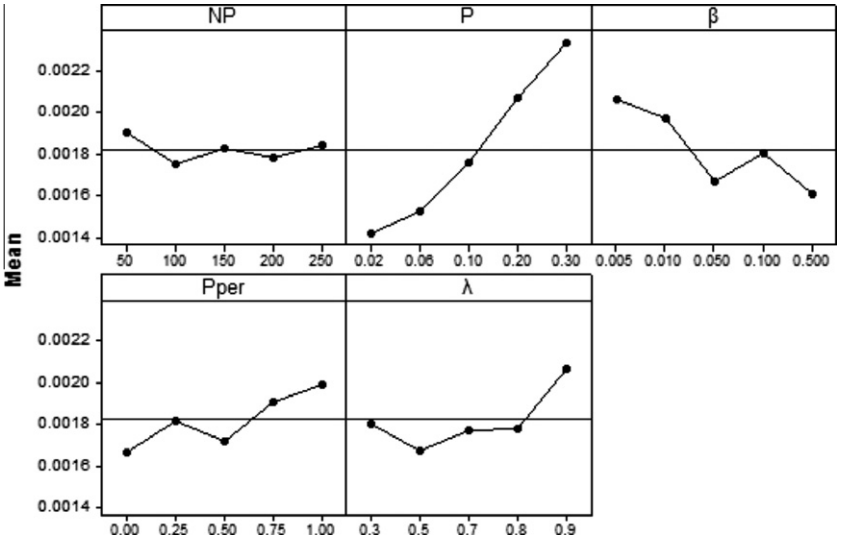

Fig. 9. Factor level trend of the HEDA for J30.

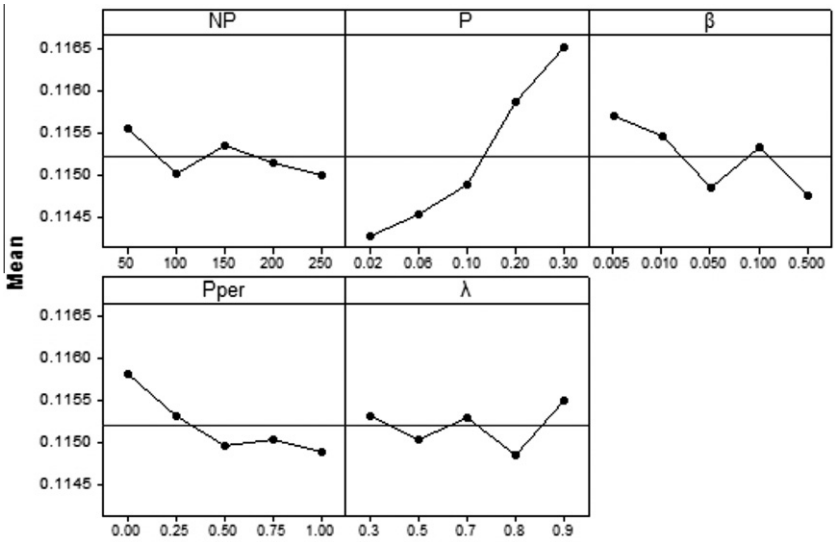

Fig. 10. Factor level trend of the HEDA for J60.

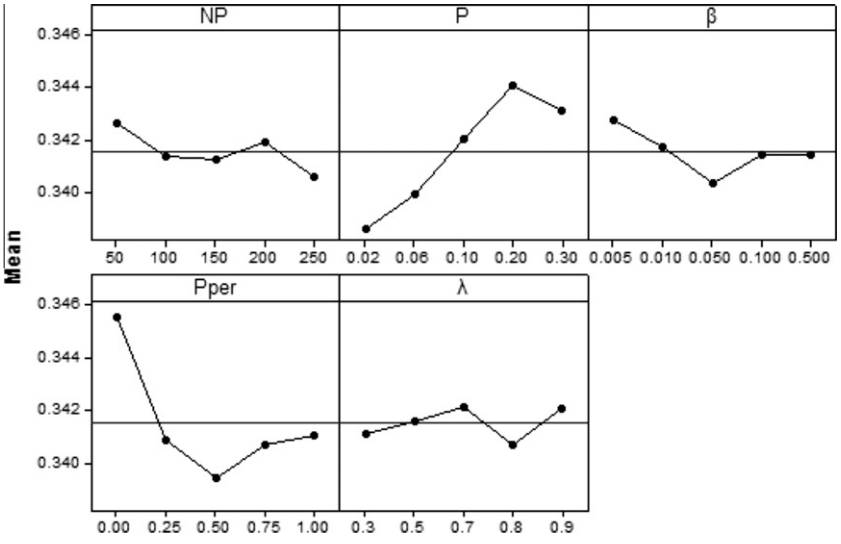

Fig. 11. Factor level trend of the HEDA for J120.

Table 3
Response table for J30.

Level	*NP*	*P*	β	*Pper*	λ
1	0.001903	0.001422	0.002065	0.001668	0.001809
2	0.001754	0.001525	0.001971	0.001821	0.001680
3	0.001831	0.001762	0.001667	0.001722	0.001777
4	0.001784	0.002073	0.001806	0.001910	0.001784
5	0.001847	0.002338	0.001610	0.001997	0.002068
Delta	0.000149	0.000916	0.000455	0.000329	0.000389
Rank	5	1	2	4	3

Table 4
Response table for J60.

Level	*NP*	*P*	β	*Pper*	λ
1	0.1156	0.1143	0.1157	0.1158	0.1153
2	0.1150	0.1145	0.1155	0.1153	0.1151
3	0.1154	0.1149	0.1148	0.1150	0.1153
4	0.1151	0.1159	0.1153	0.1150	0.1149
5	0.1150	0.1165	0.1148	0.1149	0.1155
Delta	0.0006	0.0023	0.0009	0.0009	0.0007
Rank	5	1	2	3	4

Table 5
Response table for J120.

Level	*NP*	*P*	β	*Pper*	λ
1	0.3426	0.3386	0.3428	0.3456	0.3411
2	0.3414	0.3399	0.3418	0.3409	0.3416
3	0.3413	0.342	0.3404	0.3395	0.3422
4	0.3419	0.3441	0.3414	0.3407	0.3407
5	0.3406	0.3431	0.3414	0.3411	0.3421
Delta	0.002	0.0055	0.0024	0.0061	0.0014
Rank	4	2	3	1	5

Table 6
The best combination of parameters for the HEDA.

Problem set	*NP*	*P*	β	*Pper*	λ
J30	100	0.02∗*NP*	0.5	0	0.5
J60	100 or 250	0.02∗*NP*	0.5	1	0.8
J120	250	0.02∗*NP*	0.05	0.5	0.8

lem is NP-hard. Fortunately, the average CPU time used by the HEDA is acceptable, which increases almost linearly with respect to the number of schedules of every problem set. In addition, for each set of problems, the derivation values decrease as the maximum number of schedules increases. Moreover, the variances of deviation for all the sets are very small, which means that the HEDA is very robust.

5.3. Comparisons of HEDA with existing algorithms

In this sub-section, we will compare the HEDA with some existing algorithms based on the PSPLIB data sets to further show the effectiveness of the HEDA. For data sets J30, J60, J120, the Ave.LB.-Dev values of all the algorithms are listed in Tables 8–10, respectively.

Table 8
Average deviations (%) for J30.

Algorithm	Maximum no. of schedules	
	1000	5000
Debels and Vanhoucke (2007)	0.12	0.04
Alcaraz, Maroto, and Ruiz (2004)	0.25	0.06
Valls, Ballestin, and Quintanilla (2008)	0.27	0.06
Agarwal, Colak, and Erenguc (2011)	0.13	0.1
Colak, Agarwal, and Erenguc (2006)	0.25	0.11
Debels et al. (2006)	0.27	0.11
HEDA	0.38	0.14
Nonobe and Ibaraki (2001)	0.46	0.16
Hartmann (2002)	0.38	0.22
Bouleimen and Lecocq (2003)	0.38	0.23
Coelho and Tavares (2003)	0.74	0.33
Schirmer (2000)	0.65	0.44
Kolisch and Drexl (1996)	0.74	0.52
Hartmann (1998)	1.03	0.56
Kolisch (1995)	1.44	1
Leon and Balakrishnan (1995)	2.08	1.59

Table 9
Average deviations (%) for J60.

Algorithm	Maximum no. of schedules	
	1000	5000
Debels and Vanhoucke (2007)	11.31	10.95
Valls et al. (2008)	11.56	11.1
Debels et al. (2006)	11.73	11.1
Alcaraz et al. (2004)	11.89	11.19
Agarwal et al. (2011)	11.51	11.29
Colak et al. (2006)	11.72	11.39
HEDA	11.97	11.43
Hartmann (2002)	12.21	11.7
Bouleimen and Lecocq (2003)	12.75	11.9
Nonobe and Ibaraki (2001)	12.97	12.18
Schirmer (2000)	12.94	12.58
Hartmann (1998)	13.3	12.74
Kolisch and Drexl (1996)	13.51	13.06
Coelho and Tavares (2003)	13.8	13.31
Leon and Balakrishnan (1995)	14.33	13.49
Kolisch (1995)	14.89	14.3

For data set J30, the HEDA is the 7th best with both 1,000 and 5,000 schedules among all the 16 algorithms. For each case, the gap between the HEDA and the best algorithm is very small. That is, 0.26% with 1,000 schedules, and 0.1% with 5,000 schedules. Since the makespan obtained by the HEDA is much close to the optimal bound, the HEDA is effective to solve RCPSP with small scale.

For data set J60, the HEDA is the 8th best with 1,000 and 7th best 5,000 schedules among the 16 algorithms. The gap between the HEDA and the best algorithm is 0.66% with 1,000 schedules, and 0.48% with 5,000 schedules.

For data set J120, the HEDA is the 7th best with 1,000 schedules and is the 4th best with 5,000 schedules, where the gap between

Table 7
The results for the HEDA.

Problem set	Schedules	Ave.LB.Dev (%)	Min.LB.Dev (%)	Max.LB.Dev (%)	Var.	Ave.CPU (s)	Min.CPU (s)	Max.CPU (s)
J30	1,000	0.38	0.33	0.41	5.32e-08	0.0159	0.0158	0.0160
	5,000	0.14	0.11	0.17	2.92e-08	0.0796	0.0794	0.0797
J60	1,000	11.97	11.18	12.54	6.51e-05	0.0919	0.0905	0.0935
	5,000	11.43	10.85	11.99	4.99e-05	0.4074	0.4054	0.4108
J120	1,000	35.44	33.86	36.73	1.15e-04	0.6220	0.6167	0.6304
	5,000	33.61	32.87	23.74	9.54e-05	2.8426	2.8273	2.8792

Table 10
Average deviations (%) for J120.

Algorithm	Maximum no. of schedules	
	1000	5000
Debels and Vanhoucke (2007)	33.55	32.18
Valls et al. (2008)	34.07	32.54
Debels et al. (2006)	35.22	33.1
HEDA	35.44	33.61
Alcaraz et al. (2004)	36.53	33.91
Agarwal et al. (2011)	34.65	34.15
Colak et al. (2006)	34.94	34.57
Hartmann (2002)	37.19	35.39
Bouleimen and Lecocq (2003)	42.81	37.68
Nonobe and Ibaraki (2001)	40.86	37.88
Hartmann (1998)	39.93	38.49
Schirmer (2000)	39.85	38.7
Kolisch and Drexl (1996)	41.37	40.45
Coelho and Tavares (2003)	41.36	40.46
(Leon and Balakrishnan (1995)	42.91	40.69
Kolisch (1995)	44.46	43.05

the HEDA and the best algorithm is 1.89% with 1,000 schedules, and 1.43% with 5,000 schedules. The gap decreases as the number of schedules increases.

All in all, it can be concluded that HEDA is competitive with the existing algorithms for solving RCPSP, especially for the problems with media and large scales.

6. Conclusions

In this paper, a hybrid estimation of distribution algorithm was proposed for solving the RCPSP. Different from existing methods, the HEDA adopted the statistic tool to predict the most promising area. Additionally, problem specific knowledge about the precedence constraint of the RCPSP was taken into account in developing the novel probability model updating mechanism, which made the HEDA could sample the most promising area more effectively. By using an encoding scheme based on the extended active list and a decoding scheme based on the improved serial SGS, the HEDA was applied to RCPSP conveniently. By applying the combined local search with permutation based local search and Forward–Backward improvement, the exploitation was enhanced. Based on the PSPLIB benchmarks, the Taguchi method of DOE was used to determine a suitable parameter setting of the HEDA, and the simulation results and comparisons with some existing algorithms demonstrated the effectiveness of the HEDA. Our future work is to develop the adaptive HEDA for the RCPSP and design effective HEDA for some generalized RCPSP, such as multi-mode RCPSP and stochastic RCPSP.

Acknowledgments

This research is partially supported by National Science Foundation of China (61174189, 61025018, 70871065 and 60834004), Program for New Century Excellent Talents in University (NCET-10-0505), Doctoral Program Foundation of Institutions of Higher Education of China (20100002110014), the National Key Basic Research and Development Program of China (No.2009CB320602) and National Science and Technology Major Project of China (No.2011ZX02504-008).

References

Agarwal, A., Colak, S., & Erenguc, S. (2011). A neurogenetic approach for the resource-constrained project scheduling problem. *Computers & Operations Research, 38*, 44–50.

Aickelin, U., Burke, E. K., & Li, J. (2007). An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering. *Journal of the Operational Research Society, 58*, 1574–1585.

Alba, E., & Francisco Chicano, J. (2007). Software project management with GAs. *Information Sciences, 177*, 2380–2401.

Alcaraz, J., Maroto, C., & Ruiz, R., (2004). Improving the performance of genetic algorithms for the RCPS problem. In *Proceedings of the 9th International Workshop on Project Management and Scheduling* (pp. 40–43).

Alcaraz, J., & Maroto, C. (2001). A robust genetic algorithm for resource allocation in project scheduling. *Annals of Operations Research, 102*, 83–109.

Alvarez-Valdes, R., & Tamarit, J. M. (1989). Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis. In R. Slowinski & J. Weglarz (Eds.), *Advances in project scheduling* (pp. 134–143). Amsterdam: Elsevier.

Artigues, C., Michelon, P., & Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research, 149*, 249–267.

Blazewicz, J., Lenstra, J. K., & Rinnooy, K. (1983). Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics, 5*, 11–24.

Boctor, F. F. (1996). A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. *European Journal of Operational Research, 90*, 349–361.

Bouleimen, K., & Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research, 149*, 268–281.

Brucker, P., Drexl, A., Mohring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: notation, classification, models, and methods. *European Journal of Operational Research, 112*, 3–41.

Cesar, R. M., Jr., Bengoetxea, E., Bloch, I., & Larranaga, P. (2005). Inexact graph matching for model-based recognition: evaluation and comparison of optimization algorithms. *Pattern Recognition, 38*, 2099–2113.

Cho, J. H., & Kim, Y. D. (1997). A simulated annealing algorithm for resource constrained project scheduling problems. *Journal of the Operational Research Society, 48*, 736–744.

Coelho, J., & Tavares, L., (2003). Comparative analysis of meta-heuricstics for the resource constrained project scheduling problem. In Technical report. Lisbon: *Department of Civil Engineering*, Instituto Superior Tecnico.

Colak, S., Agarwal, A., & Erenguc, S. S. (2006). Resource constrained project scheduling: a hybrid neural approach. In J. Weglarz & J. Jozefowska (Eds.), *Perspectives in Modern Project Scheduling* (pp. 297–318). Berlin: Springer.

Cooper, D. F. (1977). A note on serial and parallel heuristics for resource-constrained project scheduling. *Foundations of Control Engineering, 2*, 131–133.

Davis, E. W., & Patterson, J. H. (1975). A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management Science, 944*, 955.

Debels, D., De Reyck, B., Leus, R., & Vanhoucke, M. (2006). A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research, 169*, 638–653.

Debels, D., & Vanhoucke, M. (2007). A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Operations Research, 55*, 457–469.

Dodin, B., Elimam, A. A., & Rolland, E. (1998). Tabu search in audit scheduling. *European Journal of Operational Research, 106*, 373–392.

Drexl, A. (1991). Scheduling of project networks by job assignment. *Management Science, 37*, 1590–1602.

Hartmann, S. (1997). *Scheduling medical research experiments: an application of project scheduling methods*. Germany: University Kiel. Technique report.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics, 45*, 733–750.

Hartmann, S. (1999). *Project scheduling under limited resources: models, methods, and applications.*. Berlin: Springer Verlag.

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics, 49*, 433–448.

Jarboui, B., Eddaly, M., & Siarry, P. (2009). An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. *Computers & Operations Research, 36*, 2638–2646.

Klein, R. (2000). Project scheduling with time-varying resource constraints. *International Journal of Production Research, 38*, 3937–3952.

Kolisch, R. (1995). *Project scheduling under resource constraints: efficient heuristics for several problem classes*. Heidelberg: Springer Verlag.

Kolisch, R. (1996a). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management, 14*, 179–192.

Kolisch, R. (1996b). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research, 90*, 320–333.

Kolisch, R., & Drexl, A. (1996). Adaptive search for solving hard project scheduling problems. *Naval Research Logistics, 43*, 23–40.

Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. In J. Weglarz (Ed.), *Project scheduling, recent models, algorithms and applications* (pp. 147–178). Berlin: Kluwer.

Kolisch, R., & Sprecher, A. (1996). PSPLIB-A project scheduling problem library. *European Journal of Operational Research, 96*, 205–216.

Larranaga, P., & Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Netherlands: Springer.

Leon, V. J., & Balakrishnan, R. (1995). Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling. *OR Spectrum, 17*, 173–182.

Li, K. Y., & Willis, R. J. (1992). An iterative scheduling technique for resource-constrained project scheduling. *European Journal of Operational Research, 56*, 370–379.

Lozano, J. A., Larra aga, P., & Bengoetxea, E. (2006). *Towards a new evolutionary computation: advances in the estimation of distribution algorithms*. New York: Springer-Verlag.

Montgomery, D. C. (2005). *Design and analysis of experiments*. Arizona: John Wiley & Sons.

Nonobe, K., & Ibaraki, T. (2001). Formulation and tabu search algorithm for the resource constrained project scheduling problem. In C. C. Ribeiro & P. Hansen (Eds.), *Essays and Surveys in Metaheuristics* (pp. 557–588). Boston: Kluwer.

Palmer, C.C., & Kershenbaum, A. (1994). Representing trees in genetic algorithms. In *Proceedings of the first IEEE International conference on Evolutionary Computation* (pp. 376–384).

Saeys, Y., Degroeve, S., Aeyels, D., Van de Peer, Y., & Rouze, P. (2003). Fast feature selection using a simple estimation of distribution algorithm: A case study on splice site prediction. *Bioinformatics*, 19.

Sagarna, R., & Lozano, J. A. (2005). On the performance of estimation of distribution algorithms applied to software testing. *Applied Artificial Intelligence, 19*, 457–489.

Schirmer, A. (2000). Case-based reasoning and improved adaptive search for project scheduling. *Naval Research Logistics, 47*, 201–222.

Shaffer, L. R., Ritter, J. B., & Meyer, W. L. (1965). *The critical-path method*. New York: McGraw-Hill.

Simionescu, P. A., Beale, D., & Dozier, G. V. (2006). Teeth-number synthesis of a multispeed planetary transmission using an estimation of distribution algorithm. *Journal of Mechanical Design, 128*, 108–115.

Stinson, J. P., Davis, E. W., & Khumawala, B. M. (1978). Multiple resource-constrained scheduling using branch and bound. *IIE Transactions, 10*, 252–259.

Thomas, P. R., & Salhi, S. (1997). An investigation into the relationship of heuristic performance with network-resource characteristics. *Journal of the Operational Research Society, 48*, 34–43.

Valls, V., Ballestin, F., & Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research, 185*, 495–508.

Zhang, Q., Sun, J., Tsang, E., & Ford, J. (2003). Combination of guided local search and estimation of distribution algorithm for quadratic assignment problems. In *Genetic and Evolutionary Computation Conference*, (pp. 42–48).

Zhang, H., Li, X., Li, H., & Huang, F. (2005). Particle swarm optimization-based schemes for resource-constrained project scheduling. *Automation in Construction, 14*, 393–404.