Available online at www.sciencedirect.com
ScienceDirect
European Journal of Operational Research 193 (2009) 35–48
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/ejor

Discrete Optimization

A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling

Mohammad Ranjbar [a,*], Bert De Reyck [b,c], Fereydoon Kianfar [d]

[a] Department of Industrial Engineering, Faculty of Engineering, Ferdowsi University of Mashad, Mashad, Iran
[b] Department of Management Science and Innovation, University College London, Gower Street, London WC1E 6BT, UK
[c] Department of Management Science and Operations, London Business School, Regent's Park, London NW1 4SA, UK
[d] Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran

Received 16 December 2006; accepted 26 October 2007
Available online 5 November 2007

Abstract

We develop a heuristic procedure for solving the discrete time/resource trade-off problem in the field of project scheduling. In this problem, a project contains activities interrelated by finish-start-type precedence constraints with a time lag of zero, which require one or more constrained renewable resources. Each activity has a specified work content and can be performed in different modes, i.e. with different durations and resource requirements, as long as the required work content is met. The objective is to schedule each activity in one of its modes in order to minimize the project makespan. We use a scatter search algorithm to tackle this problem, using path relinking methodology as a solution combination method. Computational results on randomly generated problem sets are compared with the best available results indicating the efficiency of the proposed algorithm.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Project scheduling; Heuristic; Scatter search; Path relinking; Time/resource trade-offs

1. Introduction

The *resource-constrained project scheduling problem* (RCPSP) involves the non-preemptive scheduling of project activities subject to finish-start-type precedence constraints and renewable resource constraints in order to minimize the project duration. Each activity in the RCPSP can be executed in a single way. For more information on the RCPSP and solution methods, we refer to Demeulemeester and Herroelen (2002). In the multi-mode RCPSP (MRCPSP) each activity can be accomplished in one of several execution modes. Each execution mode represents an alternative combination of resource requirements of the activity and its duration. Following Slowinski (1980), three types of resources are considered in the MRCPSP: renewable, non-renewable and doubly constrained. Renewable resources are available in constrained quantities during each time period, while non-renewable resources are constrained for the overall project and doubly constrained resources are constrained per period as well as for the overall project. However, the third type of resources does not need to be considered explicitly, since they can be incorporated by considering them as a renewable as well as a non-renewable resource.

In practice, however, it is often the case that activities can be performed with different amounts of resources, resulting in different possible durations. In these cases, activities can be thought of as containing a specific work content, e.g. in terms of person-days, whereby different combinations of duration and resource requirements could be specified,

* Corresponding author. Tel.: +98 91 5559 1612.
E-mail addresses: ranjbar@um.ac.ir (M. Ranjbar), bdereyck@london.edu (B. De Reyck), kianfar@sharif.edu (F. Kianfar).

0377-2217/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2007.10.042

as long as the activity's specified work content is met. A set of allowable execution modes can then be specified for each activity, each characterized by a fixed duration and associated constant resource requirements such that the product of them should be at least equal to the activity's specified work content. The version of this problem with a single renewable resource was first introduced by De Reyck et al. (1998) as the *discrete time/resource trade-off problem* (DTRTP).

The NP-hardness of the RCPSP was shown by Blazewicz et al. (1983) as a generalization of the job shop scheduling problem. As a generalization of the RCPSP, the MRCPSP is strongly NP-hard (Kolisch, 1995). Furthermore, Demeulemeester et al. (2000) show that the DTRTP, as a generalization of the parallel machine problem, is strongly NP-hard.

In this paper, we consider the DTRTP with multiple resource types (MDTRTP). In the MDTRTP, there are multiple renewable resources, each with time/resource trade-offs. As a generalization of the DTRTP, MDTRTP is strongly NP-hard.

The MDTRTP is also a subproblem of the MRCPSP, which includes time/resource and resource/resource trade-offs for multiple renewable, non-renewable and doubly-constrained resource types. The major difference between the MDTRTP and the MRCPSP is the existence of non-renewable resources. In the MRCPSP, the mode assignment problem becomes NP-complete when there are at least two non-renewable resource types (Kolisch, 1995).

In this paper, we present a new hybrid metaheuristic algorithm based on scatter search and path relinking methods. Scatter search (SS) is an evolutionary or population-based method in which solutions are combined to yield better solutions using convex or non-convex linear combinations. SS contrasts with other evolutionary procedures such as genetic algorithms (GA), by providing unifying principles for joining solutions based on generalized path constructions in Euclidean space and by utilizing strategic designs where other approaches resort to randomization. In our SS algorithm for the MDTRTP, we use path relinking concepts to generate children from parent solutions, in the form of a new combination method. We also incorporate new strategies for diversification and intensification to enhance the search, in the form of local search and forward–backward scheduling, based on so-called reverse schedules, with the activity dependencies reversed. We also modify our procedure to tackle the MRCPSP and RCPSP.

The remainder of the paper is organized as follows. A review of the literature is given in Section 2. Section 3 describes the problem formulation and notation. The schedule representation is discussed in Section 4. The scatter search algorithm is described in detail in Section 5. Section 6 is assigned to the extension of our procedure for the MRCPSP and RCPSP. Computational results are presented in Section 7, while Section 8 is reserved for overall conclusions and suggestions for future research.

2. Literature review

Numerous exact and heuristic procedures are presented in the literature for the RCPSP and MRCPSP. Chapters 6 and 8 of the project scheduling research handbook of Demeulemeester and Herroelen (2002) give an extensive literature overview of the RCPSP and MRCPSP, respectively, and hence, it is not repeated here. In addition, Kolisch and Hartmann (2006) present a classification and performance evaluation of different heuristic and metaheuristic algorithms for the RCPSP. To the best of our knowledge, there are only two metaheuristics based on SS in the field of project scheduling: Debels et al. (2006) and Yamashita et al. (2006). The first one is a hybrid metaheuristic for the RCPSP developed on the basis of the standard structure of SS in which two elements from SS are combined by a heuristic optimization method that simulates the electromagnetism theory of physics. The second one is a metaheuristic based on SS for the resource availability cost problem in which the authors work on determining the resource availabilities and transform the problem to the RCPSP.

Contrary to the RCPSP and MRCPSP, the literature on the DTRTP and MDTRTP is sparse. De Reyck et al. (1998) present several heuristic procedures for the DTRTP based on local search and tabu search (TS). These heuristic procedures are based on the decomposition of the problem into a mode assignment phase and a resource-constrained project scheduling phase with fixed mode assignments. They use different types of memory and aspiration criteria to improve the procedure's performance. Demeulemeester et al. (2000) have developed a branch-and-bound algorithm (B&B) for the DTRTP based on the concept of activity–mode combinations, i.e. subsets of activities executed in a specific mode. At each decision point t which corresponds to the completion time of one or more activities, the algorithm evaluates feasible partial schedules PS_t (corresponding to nodes in the search tree) obtained by enumerating all *feasible maximal activity–mode combinations*. Activity–mode combinations are *feasible* if the activities can be executed in parallel in the specified mode without resulting in a resource constraint violation. They are *maximal* when no other activity can be added in one of its modes without causing a resource conflict. Each partial schedule resulting from a specific activity–mode combination is evaluated by computing a critical path-based and a resource-based lower bound.

Recently, Ranjbar and Kianfar (2007) have developed a metaheuristic procedure based on genetic algorithms for the DTRTP. They use a two-point crossover in which crossover points are determined based on the resource utilization ratio.

3. Problem formulation and notation

The objective of the MDTRTP is to schedule each activity of a project in one of its defined modes subject to precedence and resource constraints, in order to minimize the project makespan. The duration of each activity is not predetermined, but changes as a discrete non-increasing function of the amount of renewable resources assigned to it. The notations are defined in Table 1. Activities 0 and n represent the start and completion of the project respectively, with $w_{0k} = w_{nk} = 0$, $k = 1,\ldots,|R|$ and $|M_0| = |M_n| = 1$. For each activity j, each efficient mode $(d_{jm}, \mathbf{r}_{jm})$ is allowed as long as $r_{jmk} d_{jm} \geqslant w_{jk}$ for each resource type k. A mode is efficient if every other mode has a strictly higher duration or a strictly higher resource requirement for at least one resource k. Furthermore, we assume that the modes of each activity are sorted in the order of non-decreasing duration.

The MDTRTP can be formulated by introducing the decision variable y_{jmt} as follows:

$$y_{jmt} = \begin{cases} 1, & \text{if activity } j \text{ is performed in mode } m \text{ and started at time } t, \\ 0, & \text{otherwise,} \end{cases}$$

$$\text{Min} \quad \sum_{t=\text{est}_n}^{\text{lst}_n} t y_{n1t}, \tag{1}$$

$$\text{Subject to} \quad \sum_{m=1}^{|M_j|} \sum_{t=\text{est}_j}^{\text{lst}_j} y_{jmt} = 1, \quad j = 0,\ldots,n, \tag{2}$$

$$\sum_{m=1}^{|M_i|} \sum_{t=\text{est}_i}^{\text{lst}_i} (t + d_{im}) y_{imt} \leqslant \sum_{m=1}^{|M_j|} \sum_{t=\text{est}_j}^{\text{lst}_j} t y_{jmt}, \quad (i,j) \in E, \tag{3}$$

$$\sum_{j=0}^{n} \sum_{m=1}^{|M_j|} r_{jmk} \sum_{s=\max\{t-d_{jm},\text{est}_j\}}^{\min\{t-1,\text{lst}_j\}} y_{jms} \leqslant \text{AR}_k, \quad k = 1,\ldots,|R|, \ t = 1,\ldots,T, \tag{4}$$

$$y_{jmt} \in \{0,1\}, \quad j = 0,\ldots,n, \ m = 1,\ldots,|M_j|, \ t = 1,\ldots,T. \tag{5}$$

The objective function (1) minimizes the project makespan. Constraints (2) assure that exactly one mode and one start time are assigned to each activity. Constraints (3) and (4) indicate the precedence and resource constraints, respectively. Finally, constraints (5) ensure that the decision variables are binary variables.

4. Schedule representation and generation scheme

Our constructive heuristic algorithm uses a *schedule representation* to encode a project schedule and a *schedule generation scheme* (SGS) to translate the schedule representation to a schedule $S = (\mathbf{s}, \mathbf{f})$. The SGS determines how a feasible sche-

Table 1
Notations

$J = \{0,1,\ldots,n\}$	Set of activities with index j
$E = \{(i,j);\ i,j \in J\}$	Set of precedence relations
P_j	Set of total predecessors of activity j
Q_j	Set of total successors of activity j
$R = \{1,\ldots,|R|\}$	Set of renewable resources with index k
AR_k	Constant availability of renewable resource k; $k = 1,\ldots,|R|$
w_{jk}	Work content of activity j with respect to renewable resource k; $k = 1,\ldots,|R|$
$|M_j|$	Number of modes of activity j
d_{jm}	Duration of activity j in mode m; $j = 0,1,\ldots,n$; $m = 1,\ldots,|M_j|$
r_{jmk}	Resource requirement of activity j in mode m for renewable resource k; $j = 0,1,\ldots,n$, $m = 1,\ldots,|M_j|$, $k = 1,\ldots,|R|$
$\mathbf{r}_{jm} = (r_{jmk})$	Vector of resource requirements of activity j in mode m; $j = 0,1,\ldots,n$, $m = 1,\ldots,|M_j|$
$M_j = \{(d_{jm}, \mathbf{r}_{jm})\} = \{1,\ldots,|M_j|\}$	Set of possible modes for activity j; $j = 0,1,\ldots,n$; $m = 1,\ldots,|M_j|$
$d_j^{\min} = \max_{k\in R}\{\lceil w_{jk}/\text{AR}_k \rceil\}$	Minimum possible duration for activity j; $j = 0,1,\ldots,n$
$d_j^{\max} = \max_{k\in R}\{w_{jk}\}$	Maximum possible duration for activity j; $j = 0,1,\ldots,n$
$\text{est}_j = \max\left\{\left(\max_k\left\{\left\lceil \sum_{i\in P_j} w_{ik}/\text{AR}_k \right\rceil\right\}\right), \left(\max_{i\in P_j}\{\text{est}_i + d_i^{\min}\}\right)\right\}$	Earliest start time of activity j; $j = 0,1,\ldots,n$
$\text{lst}_j = \min\left\{\left(T - d_j^{\min} - \max_k\left\{\left\lceil \sum_{i\in Q_j} w_{ik}/\text{AR}_k \right\rceil\right\}\right), \left(T - \min_{i\in Q_j}\{\text{lst}_i\} - d_j^{\min}\right)\right\}$	Latest start time of activity j; $j = 0,1,\ldots,n$
$\mathbf{s} = (s_0,\ldots,s_n)$	Set of activity start times
$\mathbf{f} = (f_0,\ldots,f_n)$	Set of activity finish times
T	An upper bound on the project makespan

dule is constructed by assigning starting times to the activities, whereby the relative priorities are determined by the schedule representation.

We represent a schedule S of the MDTRTP by a double list $\mathbf{x} = (\lambda, \mu)$. The first list is an activity list $\lambda = (j_0, \ldots, j_n)$, a sequence of activities where activity j_i is the activity number with ith priority for being scheduled using the SGS, and the second list is a mode list $\mu = (m(j_0), \ldots, m(j_n))$, where $m(j_i)$ represents the mode chosen for activity j_i. So, in the ith position of the mode list the execution mode of the activity placed in the ith position of the activity list is represented. We call an activity list *precedence feasible* (PF) if every activity is positioned after all of its predecessors in the list. We also enforce the *topological order* (TO) condition introduced by Valls et al. (2003) in our activity list representation. To embed the TO-condition in a given activity list, we first schedule the activities using a serial SGS (SSGS) and then sequence them in non-increasing order of their finish times, i.e. for all i and j, if $f_i(S) > f_j(S)$, where $f_i(S)$ and $f_j(S)$ denote the finish time of activities i and j in schedule S, respectively, activity i comes before activity j in the topologically ordered activity list. The advantage of this is that, whereas several activity lists can result in the same schedule using a SSGS, each topological order corresponds to a unique schedule, except in the case of identical activity finish times. Each activity list obtained using the TO-condition is also a PF activity list for the reverse network, in which all dependencies of the activity network are reversed. For a detailed discussion of the advantages of the topological order representation, we refer to Debels et al. (2006).

We translate a representation $\mathbf{x}$ of the MDTRTP to a schedule S using a SSGS and denote its project makespan by mak($\mathbf{x}$). Kolisch (1996) shows that for the RCPSP there always is a λ^* yielding the optimum solution for a regular measure of performance when using a SSGS. Similarly for the MDTRTP, there always is a $\mathbf{x}^* = (\lambda^*, \mu^*)$ with optimum makespan using the SSGS. In each iteration of the SSGS, the activity with the highest priority is chosen and assigned the first possible starting time such that no precedence or resource constraint is violated.

5. The scatter search algorithm

5.1. General overview

Scatter search (SS) is an evolutionary method that constructs new solutions by combining existing ones in a systematic fashion. For a general introduction to SS, we refer the reader to Laguna and Marti' (2003). Fig. 1 shows the main structure of our algorithm. In the first step we generate an initial population P containing $|P|$ solutions. In the second step, we construct the reference set RefSet including RefSet_1 and RefSet_2, the former containing b_1 solutions with low makespans, the latter containing b_2 solutions with high diversity ($b = b_1 + b_2$). The solutions of RefSet_1 and RefSet_2 are called reference

1. Construct an initial population P with size of $|P|$.
2. Build the reference set using the *RefSet* building method.
3. Generate *Newsubsets* with the subset generation method. Make $P=\emptyset$.

while (*Newsubsets*$\neq\emptyset$) **do**

4. Select the next subset σ in *NewSubsets*.
5. Apply the solution combination method to σ to obtain a number of new solutions.
6. Select *nrc* of the new generated solutions in step 5, apply the intensification method over each selected solution with probability of P_{ls} and add the new obtain solutions to the P.
7. Delete σ from subsets.

end while

8. Apply the TO-condition to the best solution so far and add it to the P.
9. Reverse activity network
10. If one of the termination criteria is met, stop. Otherwise, go to 2.

Fig. 1. Scatter search procedure.

solutions. Next, we generate the NewSubsets, each of them containing two reference solutions. Subsequently, the two solutions of each subset are combined, and new solutions are generated using a path relinking algorithm which explores a set of solutions on the path between the solutions in the selected subset. From the newly generated solutions in step 5 and based on the both quality and diversity, we select *nrc* of them as children solutions. In order to improve these children solutions, we perform a local search around each of them with local search probability p_{ls} resulting in new members of P. Then, we transfer the best solution so far, after applying by the TO-condition, to the new population. In the next step, we reverse the dependencies in the activity network to perform a forward–backward improvement scheduling step (Li and Willis, 1992). More precisely, our procedure uses direct schedules, obtained from the activity network with original dependencies, to generate reverse children, obtained from the activity network with reversed dependencies, and reverse schedules to generate direct children. Steps 2–9 are repeated until the termination criterion is reached.

5.2. Construction of initial population P

Each element in the initial population is created by randomly generating activity and mode lists, constructing the corresponding schedules using a SSGS and then modifying the activity lists in order to satisfy the TO-condition.

To guarantee a diversified population of solutions, we use biased random sampling using frequency memory. When generating solutions, the probability that activity j is placed in position i in the activity list equals:

$$p_{ji} = \frac{1/n_{ji}}{\sum_{p\in \mathrm{NSP}} 1/n_{jp}}, \quad j,i = 1,\ldots,n-1, \tag{6}$$

where n_{ji} is the number of times that activity j was placed in position i in previously generated solutions and NSP denotes the set of unselected positions in the current activity list under construction. Similarly, the probability that mode m is selected for activity j equals:

$$p_{jm} = \frac{1/n_{jm}}{\sum_{q=1}^{|M_j|} 1/n_{jq}}, \quad j = 1,\ldots,n,\ m = 1,\ldots,|M_j|, \tag{7}$$

where n_{jm} is the number of times that activity j was assigned mode m in previously generated solutions.

5.3. Intensification

A local search procedure is applied over each generated schedule S with probability P_{ls}. It changes the mode m assigned to activity j two times. First, it changes mode m to mode $m+1$ if $m < |M_j|$ and then to mode $m-1$ if $m > 1$, while the other activities remain unchanged. It then generates at least $n-2$ and at most $2(n-2)$ corresponding new solutions using the SSGS. If an improved solution is obtained, we apply the TO-condition on the best found solution and restart the local search based on this solution. This process is stopped when no more improvement is obtained.

5.4. Reference set building and subset generation methods

RefSet, the set of reference solutions, includes solutions based on both quality and diversity. The construction of high-quality solutions, RefSet_1, starts with the selection of the solution in P with the lowest makespan. This solution is added to RefSet_1 as $\mathbf{x}^1$ and deleted from P. The next best solution $\mathbf{x}$ in P is chosen and added to RefSet_1 only if $D_{\min}(\mathbf{x}) \geqslant \text{th-dist}_1$, where $D_{\min}(\mathbf{x})$ is the minimum of the distances of solution $\mathbf{x}$ to the solutions currently in RefSet_1 and th-dist_1 is a threshold distance. The distance between two solution vectors $\mathbf{x}^p$ and $\mathbf{x}^q$ is calculated as

$$D(\mathbf{x}^p,\mathbf{x}^q) = \frac{1}{2}(D(\lambda^p,\lambda^q) + D(\mu^p,\mu^q)) = \frac{1}{2(n-1)}\sum_{i=0}^{n}(D(j_i^p,j_i^q) + D(m^p(j_i),m^q(j_i))), \tag{8}$$

where

$$D(j_i^p,j_i^q) = \begin{cases} 1, & \text{if } j_i^p \neq j_i^q \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad D(m^p(j_i),m^q(j_i)) = \begin{cases} 1, & \text{if } m^p(j_i) \neq m^q(j_i) \\ 0, & \text{otherwise} \end{cases}.$$

This process is repeated until b_1 elements are selected for RefSet_1. To construct diverse solutions in RefSet_2, we follow the same strategy as for RefSet_1, but with $\text{th-dist}_2 > \text{th-dist}_1$, and with $D_{\min}(\mathbf{x})$ as the minimum distance to the solutions in both RefSet_1 and RefSet_2. Therefore, both RefSet_1 and RefSet_2 contain diversified solutions, but with more emphasis on diversification in RefSet_2. When no qualified solution can be found in the population, we complete RefSet with randomly generated solutions based on the method used for building the initial population. In this case, we do not check the minimum threshold distance condition for the generated solutions.

After the RefSet construction, we generate NewSubsets. Each subset includes two solutions from RefSet_1 or one solution from RefSet_1 and the other one from RefSet_2. Therefore, since $|\text{RefSet}_1| = b_1$ and $|\text{RefSet}_2| = b_2$, then $|\text{NewSubsets}| = b_1(b_1 - 1)/2 + b_1 b_2$.

5.5. Combination method

We combine the solutions of each subset, generated from RefSet, using path relinking (Glover et al., 2000). This approach generates new solutions by exploring trajectories connecting high-quality solutions. We start from one of these solutions, called *initiating solution* ($\mathbf{x}^{\text{in}} = (\lambda^{\text{in}}, \mu^{\text{in}})$), and generate a path in the neighborhood space leading toward another solution, called *guiding solution* ($\mathbf{x}^{\text{gu}} = (\lambda^{\text{gu}}, \mu^{\text{gu}})$). The selection of $\mathbf{x}^{\text{in}}$ and $\mathbf{x}^{\text{gu}}$ is based on the makespan so that $\text{mak}(\mathbf{x}^{\text{gu}}) \leqslant \text{mak}(\mathbf{x}^{\text{in}})$ and the move direction is always from $\mathbf{x}^{\text{in}}$ towards $\mathbf{x}^{\text{gu}}$. In the combination method, we explore a set of PF activity lists obtained by moving between the activity lists of the two elements of each subset. Since λ^{in} and λ^{gu} are obtained based on the TO-condition, they are PF as well.

To ensure that the new activity lists are PF, we use the following process. Suppose we have a PF activity list $\bar{\lambda} = (j_0, \ldots, j_n)$. If we interchange activities j_i and j_p, where $p > i$, we obtain a new activity list which may not be PF. Suppose we know that all predecessors of activity j_p are located before position i. Therefore, we start from position $i + 1$ and move to the right and check at each position q if the activity at that position is a predecessor of activity j_q. If this is the case, we interchange the activities in positions q and p and continue until $q = p$.

The combination method works as follows. First we initialize $\mathbf{x}^{\text{cu}}$, the current solution between $\mathbf{x}^{\text{in}}$ and $\mathbf{x}^{\text{gu}}$, as $\mathbf{x}^{\text{cu}} = \mathbf{x}^{\text{in}}$. Then we create a set C, the set of children solutions, as follows: we find the smallest i for which $j_i^{\text{cu}} \neq j_i^{\text{gu}}$ or $m(j_i^{\text{cu}}) \neq m(j_i^{\text{gu}})$. If $m(j_i^{\text{cu}}) \neq m(j_i^{\text{gu}})$ and $j_i^{\text{cu}} = j_i^{\text{gu}}$, we update $\mathbf{x}^{\text{cu}}$ by changing $m(j_i^{\text{cu}})$ as $m(j_i^{\text{cu}}) = m(j_i^{\text{gu}})$ and add it to C; otherwise, we have $j_i^{\text{cu}} \neq j_i^{\text{gu}}$. In this case, assume j_i^{gu} is located at position p of λ^{cu}, i.e. $j_p^{\text{cu}} = j_i^{\text{gu}}$. Note that p is larger than i. Now, in order to have the same activities at position i in both λ^{cu} and λ^{gu}, we interchange the activities in positions i and p of λ^{cu}. Since the ith position of every mode list is related to j_i of the corresponding activity list, at this point, we have to exchange the modes in positions i and p of μ^{cu}. If the resulting activity list is not PF, we make it PF as explained above. The resulting solution in this stage is not considered as a candidate for the set C if $m(j_i^{\text{cu}}) \neq m(j_i^{\text{gu}})$. Thus, we change the mode of the new j_i^{cu} as $m(j_i^{\text{cu}}) = m(j_i^{\text{gu}})$ if $m(j_i^{\text{cu}}) \neq m(j_i^{\text{gu}})$ and then add the new $\mathbf{x}^{\text{cu}}$ to the set C. Note that the new added $\mathbf{x}^{\text{cu}}$ has the same activities and modes as $\mathbf{x}^{\text{gu}}$ in positions 1 to i. This process is continued until $i = n$; i.e. $\mathbf{x}^{\text{cu}} = \mathbf{x}^{\text{gu}}$. At this point, the combination of $\mathbf{x}^{\text{in}}$ and $\mathbf{x}^{\text{gu}}$ is finished and the set C contains $|C|$ solutions whose characteristics change progressively from characteristics of $\mathbf{x}^{\text{in}}$ to that of $\mathbf{x}^{\text{gu}}$.

After the combination of two elements of each subset, we select *nrc* solutions from C. For that purpose, we partition set C into subsets $C_1, \ldots, C_{\lfloor |C|/nrc \rfloor}$, apply the SSGS to all solutions of C and finally we select the best solution from each subset C_c. Before adding the selected solutions to the population, we apply the TO-condition.

5.5.1. Numerical example

An example project is shown in Fig. 2, with work content and mode information in Table 2.

We assume that there is only one renewable resource with 10 available units in each time period. To start the combination method, assume the two following solutions are given. Based on the makespan of the solutions, we determine $\mathbf{x}^{\text{in}}$ and $\mathbf{x}^{\text{gu}}$:

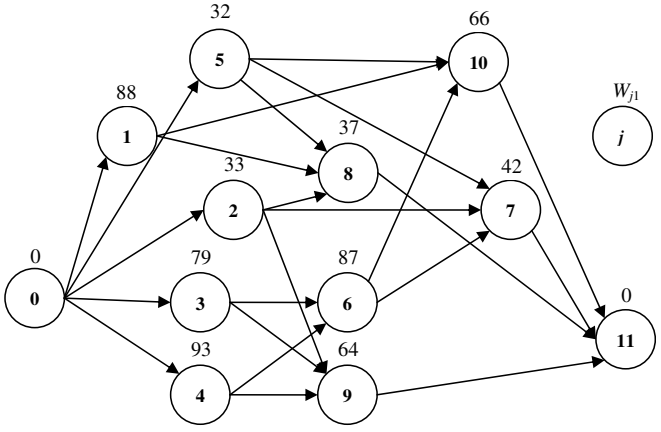

Fig. 2. Example project.

Table 2
All possible modes of the example project

Activity (j)	Work content (w_{j1})	Modes ($M_j = \{(d_{jm}, r_{jm1})\} = \{1, \ldots, m_{\lvert M_j\rvert}\}$)	$\lvert M_j\rvert$
0	0	{(0,0)}	1
1	88	{(9,10),(10,9),(11,8),(13,7),(15,6),(18,5),(22,4),(30,3),(44,2),(88,1)}	10
2	33	{(4,9),(5,7),(6,6),(7,5),(9,4),(11,3),(17,2),(33,1)}	8
3	79	{(8,10),(9,9),(10,8),(12,7),(14,6),(16,5),(20,4),(27,3),(40,2),(79,1)}	10
4	93	{(10,10),(11,9),(12,8),(14,7),(16,6),(19,5),(24,4),(31,3),(47,2),(93,1)}	10
5	32	{(4,8),(5,7),(6,6),(7,5),(8,4),(11,3),(16,2),(32,1)}	8
6	87	{(9,10),(10,9),(11,8),(13,7),(15,6),(18,5),(22,4),(29,3),(44,2),(87,1)}	10
7	42	{(5,9),(6,7),(7,6),(9,5),(11,4),(14,3),(21,2),(42,1)}	8
8	37	{(4,10),(5,8),(6,7),(7,6),(8,5),(10,4),(13,3),(19,2),(37,1)}	9
9	64	{(7,10),(8,8),(10,7),(11,6),(13,5),(16,4),(22,3),(32,2),(64,1)}	9
10	66	{(7,10),(8,9),(9,8),(10,7),(11,6),(14,5),(17,4),(22,3),(33,2),(66,1)}	10
11	0	{(0,0)}	1

$$\mathbf{x}^{\text{in}} = (\lambda^{\text{in}}, \mu^{\text{in}}) = ((0,4,3,6,5,1,2,7,9,10,8,11),(1,8,1,1,8,2,1,3,6,5,2,1)), \quad \text{mak}(\mathbf{x}^{\text{in}}) = 103,$$
$$\mathbf{x}^{\text{gu}} = (\lambda^{\text{gu}}, \mu^{\text{gu}}) = ((0,4,2,5,3,1,6,8,9,10,7,11),(1,2,8,1,7,6,3,8,6,8,2,1)), \quad \text{mak}(\mathbf{x}^{\text{gu}}) = 68.$$

First, we initialize $\mathbf{x}^{\text{cu}}$ as $\mathbf{x}^{\text{cu}} = \mathbf{x}^{\text{in}}$.

The combination of these solutions is illustrated in Fig. 3.

The first difference between $\mathbf{x}^{\text{cu}}$ and $\mathbf{x}^{\text{gu}}$ occurs at $i = 1$ for which $m^{\text{cu}}(j_1) = 8 \neq m^{\text{gu}}(j_1) = 2$ while $j_1^{\text{cu}} = j_1^{\text{gu}} = 4$. We change $m^{\text{cu}}(j_1)$ as $m^{\text{cu}}(j_1) = 2$ and add the resulting solution to set C. The next difference is related to $i = 2$ for which $j_2^{\text{cu}} = 3 \neq j_2^{\text{gu}} = 2$. As explained in the combination method, we exchange $j_2^{\text{cu}} = 3$ with $j_6^{\text{cu}} = 2$ because $j_6^{\text{cu}} = j_2^{\text{gu}} = 2$. In order to arrange μ^{cu} based on the λ^{cu}, we exchange $m^{\text{cu}}(j_2)$ and $m^{\text{cu}}(j_6)$. At this stage, λ^{cu} is not precedence feasible because activity 3 is placed after activity 6, the successor of activity 3. We make λ^{cu} precedence feasible by exchanging activities 6 and 3 in λ^{cu} and also their corresponding modes in μ^{cu}. Since $m^{\text{cu}}(j_2) = 1 \neq m^{\text{gu}}(j_2) = 8$, the resulting solution in this stage is not considered as a candidate for set C, we adjust $m^{\text{cu}}(j_2) = m^{\text{gu}}(j_2) = 8$. Now, the new $\mathbf{x}^{\text{cu}}$, with $\lambda_i^{\text{cu}} = \lambda_i^{\text{gu}}$ and $\mu_i^{\text{cu}} = \mu_i^{\text{gu}}$ for $i = 0,1,2,3$, is added to set C as a new solution. This process continues until $\mathbf{x}^{\text{cu}} = \mathbf{x}^{\text{gu}}$. It should be noted that the first and the last solution ($\mathbf{x}^{\text{in}}$ and $\mathbf{x}^{\text{gu}}$) are not considered as the elements of C. This combination results

$\mathbf{x}^{cu} = (\lambda^{cu}, \mu^{cu})$	i	λ^{cu} is PF?	Makespan	$\lvert C\rvert$
((0,4,3,6,5,1,2,7,9,10,8,11),(1,**8**,1,1,8,2,1,3,6,5,2,1))	1	yes	103	0
((0,4,**3**,6,5,1,**2**,7,9,10,8,11),(1,2,**1**,1,8,2,**1**,3,6,5,2,1))	2	yes	83	1
((0,4,2,**6**,5,1,**3**,7,9,10,8,11),(1,2,1,**1**,8,2,**1**,3,6,5,2,1))	2	no	-	1
((0,4,2,3,5,1,6,7,9,10,8,11),(1,2,**1**,1,8,2,1,3,6,5,2,1))	2	yes	-	1
((0,4,2,**3**,**5**,1,6,7,9,10,8,11),(1,2,8,**1**,**8**,2,1,3,6,5,2,1))	3	yes	105	2
((0,4,2,5,3,1,6,7,9,10,8,11),(1,2,8,**8**,1,2,1,3,6,5,2,1))	3	yes	-	2
((0,4,2,5,3,1,6,7,9,10,8,11),(1,2,8,1,**1**,2,1,3,6,5,2,1))	4	yes	73	3
((0,4,2,5,3,1,6,7,9,10,8,11),(1,2,8,1,7,**2**,1,3,6,5,2,1))	5	yes	77	4
((0,4,2,5,3,1,6,7,9,10,8,11),(1,2,8,1,7,6,**1**,3,6,5,2,1))	6	yes	67	5
((0,4,2,5,3,1,6,**7**,9,10,**8**,11),(1,2,8,1,7,6,3,**3**,6,5,**2**,1))	7	yes	69	6
((0,4,2,5,3,1,6,8,9,10,7,11),(1,2,8,1,7,6,3,**2**,6,5,3,1))	7	yes	-	6
((0,4,2,5,3,1,6,8,9,10,7,11),(1,2,8,1,7,6,3,8,6,**5**,3,1))	9	yes	70	7
((0,4,2,5,3,1,6,8,9,10,7,11),(1,2,8,1,7,6,3,8,6,8,**3**,1))	10	yes	69	8
((0,4,2,5,3,1,6,8,9,10,7,11),(1,2,8,1,7,6,3,8,6,8,2,1)	-	yes	69	8

Fig. 3. Illustration of the combination method.

in eight new solutions. If we assume $nrc = 2$, we have to select one solution from solutions $1,\ldots,4$ and the other one from solutions $5,\ldots,8$. Based on the makespan of solutions, we select solution 3 with makespan 73 and solution 5 with makespan 67. Before adding these two new solutions to the P, we apply the TO-condition to both of them resulting in the two following solutions:

$$\{(11,8,10,9,7,6,3,2,1,5,4,0),(1,2,5,6,3,1,1,8,2,8,2,1)\}, \quad \text{makespan} = 73,$$
$$\{(11,8,10,9,7,6,3,2,1,5,4,0),(1,2,5,6,3,1,7,8,6,1,2,1)\}, \quad \text{makespan} = 67.$$

These new solutions along other elements of new population will be evaluated in the next iteration of SS procedure based on the network with reverse dependencies.

5.6. Termination criteria

The procedure is terminated when (a) the specified time limit is exceeded, or (b) a solution is encountered with a makespan equal to a known lower bound. We use as a lower bound the maximum of the critical path-based lower bound LB_0 and the resource-based lower bound LB_r. LB_0 is obtained by calculating the critical path in the activity network where each activity is assigned its shortest feasible mode, taking into account the resource availabilities. LB_r is computed as $LB_r = \max_k \left\{ \left\lceil \sum_{j \in J} w_{jk} \Big/ AR_k \right\rceil \right\}$. Since d_{jm} is discrete, if for a specific activity j and resource k, $w'_{jk} = \min_{m=1}^{|M_j|}\{d_{jm} r_{jmk}\}$ exceeds w_{jk}, we use w'_{jk} for computing LB_r rather than w_{jk} itself to reach to a stronger lower bound.

6. Modification of the procedure for the RCPSP and MRCPSP

Although we have designed our procedure for the MDTRTP and DTRTP, it can be modified to tackle RCPSP and MRCPSP. For both RCPSP and MRCPSP, we change the selection strategy of the newly generated solutions from the combination of two reference solutions. Contrary to the MDTRTP and DTRTP, for which we apply the SSGS over all generated solutions to select the best solution from each C_c, where $c = 1,\ldots,\lfloor |C|/nrc \rfloor$, we select a solution randomly from each C_c and then apply the SSGS to the selected solutions. In the combination method, the higher the number of activities, the higher the number of solutions generated from the combination of two reference solutions. On the other hand, the common termination criterion for the RCPSP and MRCPSP is the maximum number of times SSGS (max-sch) is used. Therefore, this new selection strategy is used. For the same reason, we also exclude the local search from our procedure for both RCPSP and MRCPSP.

6.1. Modification of the procedure for the RCPSP

Since the MDTRTP is a generalization of the RCPSP, to tackle the RCPSP, it is sufficient only to consider $|M_j| = 1$ for all $j \in J$ where the single execution mode of each activity is defined by the input parameters. To measure the distance of two activity lists of a RCPSP instance, we use the sum of the absolute values of the component-wise differences divided by the number of activities (Debels et al., 2006).

6.2. Modification of the procedure for the MRCPSP

The most important difference here is the existence of non-renewable resources. Each activity j executed in mode m uses a total amount of nr_{jml} units of the non-renewable resource $l \in \boldsymbol{NR} = \{1,\ldots,|\boldsymbol{NR}|\}$. For each non-renewable resource l, the overall availability for the entire project is represented by ANR_l.

Before starting the SS procedure, we apply a pre-processing procedure over the project data, in order to reduce the search space. We follow the pre-processing procedure introduced by Sprecher et al. (1997) that consists of excluding non-executable modes, inefficient modes and redundant non-renewable resources. A mode is non-executable if its execution would violate the renewable or non-renewable resources constraints. A mode is called inefficient if both its duration and resource requirement are not smaller than those of another mode of the same activity. A non-renewable resource is called redundant if the sum of the maximal requirements for this resource does not exceed its availability.

The SSGS transforms each schedule representation $\mathbf{x}$ into a feasible solution if only renewable resources are taken into account. If non-renewable resources are also considered the solution may be infeasible, with respect to the non-renewable resource constraints. Since finding a feasible solution, when there is more than one non-renewable resource, is an NP-complete problem, we allow solutions which are infeasible with respect to the non-renewable resources but feasible with respect to precedence and renewable resource constraints. However, we penalise them using the evaluation function defined by Alcaraz et al. (2003) in their genetic algorithm.

We also allow infeasible solutions to enter to the RefSet, but for transferring the best found solutions to the next population, we always consider feasible solutions. However, it is worthwhile to consider infeasible solutions as the reference solutions because the combinations of some of them may generate elite feasible solutions.

7. Computational results

7.1. Benchmark problem sets

We have coded the procedure in Visual C++ 6.0 and performed all computational experiments on a PC Pentium IV 3 GHz processor with 1024 MB of internal memory. We have validated the procedure on four datasets, for the DTRTP, MDTRTP, RCPSP, and MRCPSP.

For both RCPSP and MRCPSP, we used the PSPLIB (Kolisch and Sprecher, 1997). The RCPSP dataset consists of four sets of instances: J30, J60, J90 and J120 that contain problem instances of 30, 60, 90 and 120 activities, respectively. Sets J30, J60 and J90 contain 480 instances while set J120 contains 600 instances. For the MRCPSP, we used sets J10, J12, J14, J16, J18 and J20 of the PSPLIB. These sets contain 640 instances each, with 10, 12, 14, 16, 18 and 20 activities, three modes per activity, two renewable resources and two non-renewable resources.

The DTRTP dataset was generated by Demeulemeester et al. (2000), and includes 5250 instances with 10–30 activities. We generated a dataset for MDTRTP based on the DTRTP dataset by adding renewable resource types. The work content of every activity for each renewable resource is between 10 and 100 and the resource availability for every renewable resource varies from 10 to 50 in steps of 10.

We solve several versions of each instance using a different restriction on the number of modes, denoted $|M|$, which varies from 1 to 6 to a version with an unlimited number of modes. The modes are generated in line with De Reyck et al. (1998) as follows: The procedure generates the first mode ($m = 1$) for activity j with duration d_{j1} and resource requirement $r_{j1k} = \lceil w_{jk}/d_{j1} \rceil$ with $d_{j1} = \max_k\{\max\{\lfloor\sqrt{w_{jk}}\rfloor, \lceil w_{jk}/\mathrm{AR}_k\rceil\}\}$ when the number of modes is unrestricted and $d_{j1} = \max_k\{\lceil w_{jk}/\mathrm{AR}_k\rceil\}$ when it is restricted. Then the procedure generates the second mode with duration $d_{j1} + 1$ and corresponding resource requirements. This new mode is accepted as the second mode if at least one of the resource requirements is different. This mode generation process continues until the desired number of modes is reached or no more modes are left.

7.2. Parameters settings

Using fine tuning, we set the values of the number of children and the probability of local search as $nrc = 2$ and $P_{\mathrm{ls}} = 0.02$, respectively. The diversity thresholds th-dist$_1$ and th-dist$_2$ are set to 0.4 and 0.6, respectively, for the DTRTP, MDTRTP and MRCPSP. The values of these parameters for the RCPSP are as th-dist$_1 = 1$ and th-dist$_2 = 2.2$. The size of initial population is set to $|P| = 10\,\mathrm{TL}$ for the DTRTP and MDTRTP, where TL denotes the allocated time limit. For the RCPSP and MRCPSP, the size of initial population is set to $|P| = 10b$. The values of b_1 and b_2 are determined based on the settings in Tables 3 and 4. We investigated the results of the DTRTP and MDTRTP for three time limits, namely 1, 10 and

Table 3
Tuned values of parameters b_1 and b_2 for the DTRTP and MDTRTP

TL (seconds)	b_1	b_2
1	4	2
10	7	3
100	18	9

Table 4
Tuned values of the parameters b_1 and b_2 for the RCPSP and MRCPSP

Parameter	max-sch	RCPSP				MRCPSP
		J30	J60	J90	J120	J10, J12, J14, J16, J18, J20
b_1	1000	4	4	3	4	4
	5000	7	6	8	7	8
	50,000	21	22	18	21	–
b_2	1000	2	4	2	2	2
	5000	7	5	4	5	5
	50,000	13	15	17	14	–

100 seconds. The RCPSP was tested for max-sch=1000, 5000 and 50,000, and the MRCPSP for max-sch = 1000 and 5000, in line with the results in the literature. Since the range of number of activities is small for the DTRTP, MDTRTP and MRCPSP, the results are not sensitive to the values b_1 and b_2, but for the RCPSP, b_1 and b_2 should be adjusted for both max-sch and n.

7.3. Comparative computational results

7.3.1. DTRTP

The performance of our procedure for the DTRTP on the 5520 instances of the DTRTP dataset is summarized in Table 5. We report the total sum of the makespan of each obtained solution and the average and maximal deviation from the best known solution, which is obtained using all procedures presented in this paper as well as a long run (1000 seconds) of the branch-and-bound procedure. 5130 of those best known solutions, i.e. about 98%, are known to be optimal. Also given are the number of times the best known solution is obtained, the average and maximal deviation with respect to the lower bound, and the number of problems solved to optimality. Also, the average number of RCPSP instances solved and the average and median values for the required CPU time are reported. In order to compare our results with the available procedures in the literature, we ran the TS procedure of De Reyck et al. (1998), the B&B procedure of Demeulemeester et al. (2000) and the GA procedure of Ranjbar and Kianfar (2007). In the B&B procedure, the CPU time sometimes runs over the time limit because in this algorithm, the time is only checked when it backtracks to a lower level in the search tree.

The results show that the B&B dominates the heuristics, and that the SS procedure is the best among the heuristics. A more detailed analysis can be found in Tables 6 and 7. Although Tables 6 and 7 show that when the number of modes is limited, the B&B outperforms the SS, GA and TS procedures, when the number of modes is unlimited, the SS procedure performs best, particularly when the number of activities increases and CPU time limit is decreased (indicated by bold numbers in the tables). This trend becomes even more clear on datasets with larger number of activities. Since the B&B code was only designed for at most 32 activities, we could not test it on larger instances. Additionally, despite the fact that 98% of the solutions have a known optimal solution, our procedure found improved best-known solutions for 10 instances.

Table 5
Comparative results for the DTRTP

	CPU time limit (seconds)											
	1				10				100			
	B&B	SS	GA	TS	B&B	SS	GA	TS	B&B	SS	GA	TS
Sum	355,907	356,353	356930	357,749	355,477	355,528	355,631	356,099	355,108	355,185	355,411	355,669
Average deviation best solution (%)	0.36	0.40	0.65	0.88	0.16	0.17	0.20	0.32	0.04	0.06	0.11	0.20
Maximal deviation best solution (%)	35.30	11.11	33.02	84.00	24.32	7.13	8.23	9.09	17.65	4.32	5.65	7.14
Best solution	4780 (91%)	4376 (83%)	4253 (81%)	4134 (79%)	4950 (94%)	4802 (91%)	4726 (90%)	4497 (86%)	5168 (98%)	5048 (96%)	4883 (93%)	4669 (89%)
Average deviation LB (%)	2.73	2.77	3.06	3.26	2.53	2.51	2.55	2.69	2.40	2.41	2.47	2.56
Maximal deviation LB (%)	43.75	42.86	42.86	91.67	42.86	42.86	44.17	45.86	42.86	42.86	44.17	45.86
Optimal	4772 (91%)	4361 (83%)	4250 (81%)	4134 (79%)	4928 (94%)	4773 (91%)	4673 (89%)	4489 (85%)	5107 (97%)	4961 (94%)	4779 (91%)	4633 (88%)
Average RCPSP	–	54,084	56,719	11,643	–	461,613	53,348	39,710	–	4,102,381	5,171,470	73,835
Average CPU time	0.21	0.51	0.62	0.46	0.72	4.93	6.74	2.82	3.87	48.11	51.97	11.85
Maximum CPU time	57.69	1	1	1.22	57.61	10.02	10.01	10.25	100.28	100.14	100.05	100.22

Table 6
Average percent deviation from best solution for 1 second CPU time

	Number of modes											
	Limited (number of activities)						Unlimited (number of activities)					
	10	15	20	25	30	Global	10	15	20	25	30	Global
B&B	0.01	0.03	0.03	0.01	0.12	0.06	0.81	1.74	1.68	3.34	3.56	2.22
SS	0.03	0.04	0.09	0.27	0.28	0.14	**0.67**	**1.04**	1.93	**2.92**	**3.46**	**2.00**
GA	0.05	0.09	0.16	0.41	0.45	0.23	0.73	1.45	2.75	5.00	6.11	3.21
TS	0.06	0.13	0.22	0.54	0.69	0.33	0.79	1.80	3.41	6.90	8.22	4.22

Table 7
Average percent deviation from best solution for 10 seconds CPU time

	Number of modes											
	Limited						Unlimited					
	Number of activities						Number of activities					
	10	15	20	25	30	Global	10	15	20	25	30	Global
B& B	0.01	0.01	0.01	0.06	0.05	0.03	0.05	0.30	0.66	1.65	2.20	0.97
SS	0.01	0.01	0.03	0.09	0.10	0.05	0.13	0.44	0.98	**1.48**	**1.51**	**0.92**
GA	0.01	0.02	0.06	0.14	0.16	0.08	0.23	0.63	1.21	1.97	2.09	1.22
TS	0.01	0.04	0.09	0.23	0.24	0.12	0.35	0.80	1.55	2.33	2.64	1.53

7.3.2. MDTRTP

To the best of our knowledge, no results have been published yet on the MDTRTP. We present the results of the SS and GA procedures on the MDTRTP dataset with 1050 instances. The results of Table 8 reveal that the SS procedure outperforms the GA procedure in all cases.

7.3.3. RCPSP

Comparative results for the RCPSP are available for instances sets J30, J60 and J120 in the literature. Tables 9–11 display the rank of our procedure among 10 best heuristics to date based on the results presented by Kolisch and Hartmann (2006). The comparison is made for values of 1000, 5000 and 50,000 as the limits of the maximum number of schedules. For the J30 set, the results are given in terms of average percent deviation from the makespan of the optimal solution. For the other sets, the average percent deviation from the critical path-based lower bound is used as a measure of performance, since many optimal solutions are unknown. In the following tables, the heuristics are sorted for the case of max-

Table 8
Comparative results for the MDTRTP

	CPU time (seconds)					
	1		10		100	
	GA	SS	GA	SS	GA	SS
Sum	111,030	110,867	110,263	110,177	110,054	109,996
Average deviation best solution (%)	0.91	0.77	0.25	0.17	0.07	0.01
Maximal deviation best solution (%)	18.25	16.07	6.12	4.72	3.06	2.56
Best solution	693 (66%)	762 (73%)	915 (87%)	962 (92%)	998 (95%)	1040 (99%)
Average deviation LB (%)	11.61	9.35	9.09	8.70	8.57	8.52
Maximal deviation LB (%)	48.21	48.21	48.21	48.21	48.21	48.21
Average RCPSP	41,731	40,508	376,132	360,989	462,2218	4,471,061
Average CPU time	0.82	0.80	8.24	7.93	82.65	79.65
Maximum CPU time	1.01	1.02	10.01	10.02	100.07	100.11

Table 9
Comparative results for J30

Author (year)	max-sch		
	1000	5000	50,000
Ranjbar et al. (this work)	0.10	0.03	0.00
Kochetov and Stolyar (2003)	0.10	0.04	0.00
Debels et al. (2006)	0.27	0.11	0.01
Valls et al. (2003)	0.27	0.06	0.02
Valls et al. (2005)	0.34	0.20	0.02
Alcaraz et al. (2004)	0.25	0.06	0.03
Alcaraz and Maroto (2001)	0.33	0.12	–
Tormos and Lova (2003)	0.25	0.13	0.05
Nonobe and Ibaraki (2002)	0.46	0.16	0.05
Tormos and Lova (2001)	0.30	0.16	0.07

Table 10
Comparative results for J60

Author (year)	max-sch		
	1000	5000	50,000
Ranjbar et al. (this work)	11.59	11.07	10.64
Debels et al. (2006)	11.73	11.10	10.71
Valls et al. (2003)	11.56	11.10	10.73
Kochetov and Stolyar (2003)	11.71	11.17	10.74
Valls et al. (2005)	12.21	11.27	10.74
Alcaraz et al. (2004)	11.89	11.19	10.84
Hartmann (2002)	12.21	11.70	11.21
Hartmann (1998)	12.68	11.89	11.23
Tormos and Lova (2003)	11.88	11.62	11.36
Alcaraz and Maroto (2001)	12.57	11.86	–

Table 11
Comparative results for J120

Author (year)	max-sch		
	1000	5000	50,000
Valls et al. (2003)	34.07	32.54	31.24
Ranjbar et al. (this work)	35.08	33.24	31.49
Alcaraz et al. (2004)	36.53	33.91	31.49
Debels et al. (2006)	35.22	33.10	31.57
Valls et al. (2005)	35.39	33.24	31.58
Kochetov and Stolyar (2003)	34.74	33.36	32.06
Valls et al. (2005)	35.18	34.02	32.18
Hartmann (2002)	37.19	35.39	33.21
Tormos and Lova (2003)	35.01	34.41	33.71
Merkle et al. (2002)	–	35.43	–

sch = 50,000. As a tie-breaker, the results for 5000 and then 1000 schedules are used in sorting. Tables 9–11 reveal that our procedure outperforms the other heuristics for J30, J60 and is ranked second for J120. Furthermore, our procedure improved the best-known solutions for seven instances of J90 and 32 instances of J120.

7.3.4. MRCPSP

We compare the performance of our procedure for the MRCPCP using J10, J12, J14, J16, J18 and J20 of the PSPLIB. We compare our procedure with the genetic algorithm of Alcaraz et al. (2003) and the simulated annealing of Jozefowska et al. (2001), which are the two best heuristics for the MRCPSP. The comparative results, shown as the average percent deviation from optimal solution, for max-sch = 5000, are presented in Table 12. The results reveal that our procedure outperforms the two other heuristics.

7.4. Impact of the local search

Fig. 4 indicates the impact of the local search on the DTRTP results by changing P_{ls}. The results are obtained based on the average percent deviation from the best found solutions for all instances of the DTRTP when the allowed CPU time limit is 10 seconds. The results reveal that 0.02 is the best tuned values for P_{ls}.

Table 12
Comparative results for MRCPSP

Author (year)	Instances set					
	J10	J12	J14	J16	J18	J20
Ranjbar et al. (this work)	0.18	0.65	0.89	0.95	1.21	1.64
Alcaraz et al. (2003)	0.24	0.73	1.00	1.12	1.43	1.91
Jozefowska et al. (2001)	1.16	1.73	2.6	4.07	5.52	6.74

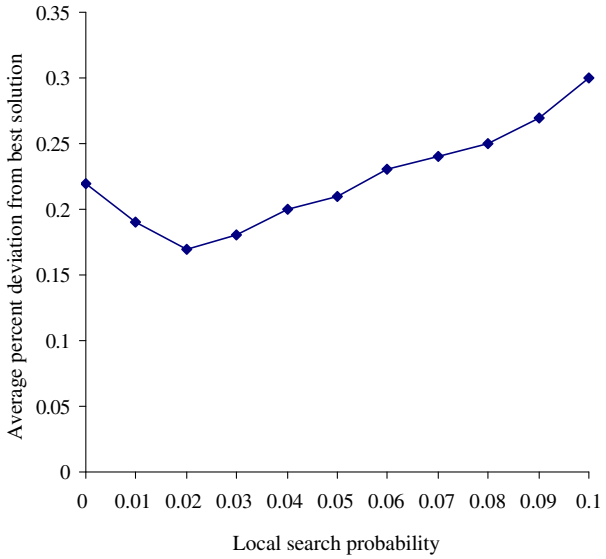

Fig. 4. The effect of the local search on the results of the DTRTP.

8. Conclusions

In this paper, we present a metaheuristic algorithm for solving the multi-resource discrete time/resource trade-off problem in project scheduling, based on scatter search and path relinking. We also modify our procedure for the DTRTP, RCPSP and MRCPSP. The performance of the algorithm is tested on four datasets, which show that our procedure in most cases outperforms the other available heuristics presented in the literature. Also, our procedure gives better results compared to the only available branch-and-bound method for the DTRTP when the number of execution modes for each activity is unlimited, the CPU time limit is small (less than 10 seconds) and the number of activities is large (more than 25).

Acknowledgements

We would like to acknowledge the office of international and scientific cooperation of Sharif University of Technology for the financial support and London Business School for providing a visiting research period for one of the authors. The authors are also indebted to the anonymous referees for useful comments and suggestions.

References

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

Alcaraz, J., Maroto, C., Ruiz, R., 2003. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. Journal of the Operational Research Society 54, 614–626.

Alcaraz, J., Maroto, C., Ruiz, R., 2004. Improving the performance of genetic algorithms for the RCPS problem. In: Proceedings of the Ninth International Workshop on Project Management and Scheduling, Nancy, pp. 40–43.

Blazewicz, J., Lemstra, J.K., Rinnooy Kan, A.H.G., 1983. Scheduling subject to resource constraints: Classification and complexity. Discrete Applied Mathematics 5, 11–24.

De Reyck, B., Demeulemeester, E., Herroelen, W., 1998. Local search methods for the discrete time/resource trade-off problem in project networks. Naval Research Logistic Quarterly 45, 553–578.

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.

Demeulemeester, E., DeReyck, B., Herroelen, W., 2000. The discrete time/resource trade-off problem in project networks: A branch and bound approach. IIE Transactions 32, 1059–1069.

Demeulemeester, E., Herroelen, W., 2002. Project Scheduling: A Research Handbook. Kluwer Academic Publishers.

Glover, F., Laguna, M., Marti', R., 2000. Fundamentals of scatter search and path relinking. Control and Cybernetics 39, 653–684.

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

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

Jozefowska, J., Mika, M., Rozycki, R., Waligora, G., Weglarz, J., 2001. Simulated annealing for multi-mode resource constrained project scheduling. Annals of Operations Research 102, 137–155.

Kochetov, Y., Stolyar, A., 2003. Evolutionary local search with variable neighbourhood for the resource constrained project scheduling problem. In: Proceedings of the Third International Workshop of Computer Science and Information Technologies, Russia.

Kolisch, R., 1995. Project Scheduling Under Resource Constraints – Efficient Heuristics for Several Problem cases. Physica-Verlag.

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

Kolisch, R., Sprecher, A., 1997. PSPLIB – A project scheduling library. European Journal of Operational Research 96, 205–216.

Kolisch, R., Hartmann, S., 2006. Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research 174, 23–37.

Laguna, M., Marti', R., 2003. Scatter Search – Methodology and Implementations in C. Kluwer Academic Publishers, Boston.

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

Merkle, D., Middendorf, M., Schmeck, H., 2002. Ant colony optimization for resource-constrained project scheduling. IEEE Transactions on Evolutionary Computation 6, 333–346.

Nonobe, K., Ibaraki, T., 2002. Formulation and tabu search algorithm for the resource constrained project scheduling problem. In: Ribeiro, C.C., Hansen, P. (Eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, pp. 557–588.

Ranjbar, M., Kianfar, F., 2007. Solving the discrete time/resource trade-off problem with genetic algorithms. Applied Mathematics and Computation 191, 451–456.

Slowinski, R., 1980. Two approaches to problems of resource allocation among project activities: A comparative study. Journal of the Operational Research Society 31, 711–723.

Sprecher, A., Hartmann, S., Drexl, A., 1997. An exact algorithm for project scheduling with multiple modes. OR Spektrum 19, 195–203.

Tormos, P., Lova, A., 2001. A competitive heuristic solution technique for resource-constrained project scheduling. Annals of Operations Research 102, 65–81.

Tormos, P., Lova, A., 2003. An efficient multi-pass heuristic for project scheduling with constrained resources. International Journal of Production Research 41 (5), 1071–1086.

Valls, V., Quintanilla, M.S., Ballestin, F., 2003. Resource-constrained project scheduling: A critical reordering heuristic. European Journal of Operational Research 165, 375–386.

Valls, V., Ballestin, F., Quintanilla, M.S., 2005. Justification and RCPSP: A technique that pays. European Journal of Operational Research 165, 375–386.

Yamashita, D.S., Armentano, V.A., Laguna, M., 2006. Scatter search for project scheduling with resource availability cost. European Journal of Operational Research 169, 623–637.