Int. J. Production Economics 117 (2009) 302–316

Contents lists available at ScienceDirect

Int. J. Production Economics

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

An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes

Antonio Lova [a], Pilar Tormos [a,*], Mariamar Cervantes [b,c], Federico Barber [b]

[a] DEIOAC, Technical University of Valencia, Spain
[b] DSIC, Technical University of Valencia, Spain
[c] University of La Sabana, Colombia

ARTICLE INFO

Article history:
Received 18 February 2008
Accepted 12 November 2008
Available online 20 November 2008

Keywords:
Project management and scheduling
Renewable and non-renewable resources
Genetic Algorithms
Multimode forward–backward improving method

ABSTRACT

Multi-mode Resource Constrained Project Scheduling Problem (MRCPSP) aims at finding the start times and execution modes for the activities of a project that optimize a given objective function while verifying a set of precedence and resource constraints. In this paper, we focus on this problem and develop a hybrid Genetic Algorithm (MM-HGA) to solve it. Its main contributions are the mode assignment procedure, the fitness function and the use of a very efficient improving method. Its performance is demonstrated by extensive computational results obtained on a set of standard instances and against the best currently available algorithms.

1. Introduction

Due to steadily shortening product life cycles, globalization of markets and decreasing profit margins, industrial projects have to be realized facing tight time and resource constraints. Within the planning process, the scheduling of jobs necessary for successfully completing a project as early as possible is a major task which is challenging and mathematically complex as soon as resource constraints are explicitly considered. Most manufacturing systems comprise resources, such as labour, machines and support equipment. Labour and work centres are usually the constraining resources when creating a schedule and their cost generally represents the major part of the production cost. Therefore, activity sequencing and resource allocation should be coordinated carefully and optimized jointly in order to optimize system performance, as specified by one or more measures (Kadrou and Najid, 2006).

Indeed, applications of project scheduling can be found in diverse contexts such as construction engineering, software development, research and development projects, etc. Moreover, project scheduling is important for make-to-order companies where the capacities have been cut down in order to meet lean management concepts (Brucker et al., 1999). Project scheduling problems consist of activities, resources, precedence relations and performance measures (Slowinski et al., 1994). When the capacity of resources is limited the resulting scheduling problems are known as Resource-Constrained Project Scheduling Problems (RCPSP).

The RCPSP involves the scheduling of the activities (or jobs) of a project in order to minimize its total duration subject to precedence relations and constant availability constraints on the required set of renewable resources (as machines or manpower). In this case each activity may be characterized by a unique duration and a singular collection of resource requirements that have to be available each time period the activity is being executed. A more general version of this problem is the Multimode Resource-Constrained Project Scheduling Problem (MRCPSP) where activity duration is a discrete function of renewable and non-renewable resources. An example of the latter resource

* Corresponding author. Tel.: +34 963877007; fax: +34 963877499.
E-mail addresses: allova@eio.upv.es (A. Lova), ptormos@eio.upv.es (P. Tormos), mcervantes@dsic.upv.es, maria.cervantes@unisabana.edu.co (M. Cervantes), fbarber@dsic.upv.es (F. Barber).

0925-5273/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ijpe.2008.11.002

category is money if the budget of the project is limited, that is, the availability of this resource within the entire project is limited.

The problem is one of the most general and most difficult project scheduling problems and as a generalization of the well-known job shop problem it belongs to the class of the NP-hard problems (Blazewicz et al., 1983). Therefore, in practice, heuristic algorithms to generate near-optimal schedules for large and highly constrained projects are needed.

This paper proposes a new heuristic approach for the MRCPSP. It is a hybrid Genetic Algorithm (GA) that uses a powerful local search method to improve the solutions provided by the GA. The main contributions are the mode assignment of the initial population, a new fitness function and the design of a very efficient improving method. Concretely, the improving method proposed in Tormos and Lova (2001) for the RCPSP has been extended to the multimode case. The consideration of multiple modes for activities requires this extension of the method that in addition to the change of the scheduling time usually implies the change of the execution mode looking for a new optimized position for each activity. The resulting method is applied to each feasible schedule drastically reducing its duration.

The remainder of the paper is organized as follows: Section 2 is devoted to problem formulation and to review the state-of-the-art. In Section 3 the new heuristic solution technique is described and the procedure to reduce the project completion time of feasible schedules is developed. The performance of the new heuristic approach is evaluated against the best heuristic and metaheuristic methods published in the literature through a wide computational experience detailed in Section 4. Finally, the main conclusions and directions of future research are drawn in Section 5.

2. Problem formulation and solving procedures

The problem considered in this paper involves the scheduling of J ($j=1,\ldots,J$) activities that may have more than one execution mode and renewable as well as non-renewable resource constraints exist. The amount of a non-renewable resource (R_{NR}) that can be consumed during the project is limited. Different activity execution modes ($m=1,\ldots,M$) require different amounts or different types of resources and represent alternative ways of realising an activity. Each activity mode thus specifies a non-preemptable unique activity duration and a set of requirements for both renewable and non-renewable resources. The objective is to obtain a feasible start/finish time (S_j/F_j) of the activities in order to minimize the project completion time.

To model the MRCPSP we assume that activities are topologically ordered, i.e. each activity j has an activity number that is larger than the number of all its immediate predecessors $i \in P_j$. We also assume the existence of a unique dummy start and finish activity, $j=1$ and $j=J$, each only realizable in a single mode associated with zero duration and zero resource demand, respectively. For all other activities we assume that modes are sorted in the order of non-decreasing duration. Each activity j and each mode m has an integer duration (d_{jm}) and uses/consumes a discrete amount of renewable/non-renewable resources (r_{jmk}). Given an upper bound T of the project's makespan (e.g. the sum of the maximum activities duration), the earliest and latest finish times, EFT_j and LFT_j, can be calculated performing a traditional forward pass assigning to each activity its shortest mode and a traditional backward pass after assigning $LFT_J = T$. Defining the variables of the model as binary variables x_{jmt} which equal 1 if activity j is scheduled in mode m ($1 \leqslant m \leqslant M_j$) to finish at t, and 0 otherwise, the problem can be formulated as follows (Talbot, 1982):

$$\text{Minimize} \quad \sum_{m=1}^{M_j} \sum_{t=EFT_j}^{LFT_j} t x_{jmt} \tag{1}$$

$$\sum_{m=1}^{M_j} \sum_{t=EFT_j}^{LFT_j} x_{jmt} = 1, \quad j = 1,\ldots,J \tag{2}$$

$$\sum_{m=1}^{M_i} \sum_{t=EFT_i}^{LFT_i} t x_{imt} \leqslant \sum_{m=1}^{M_j} \sum_{t=EFT_j}^{LFT_j} (t - d_{jm}) x_{jmt}, \quad j = 2,\ldots,J, \ i \in P_j \tag{3}$$

$$\sum_{j=2}^{J-1} \sum_{m=1}^{M_j} r_{jmk} \sum_{\tau=t}^{t+d_{jm}-1} x_{jm\tau} \leqslant R_k, \quad k \in R, \ t = 1,\ldots,T \tag{4}$$

$$\sum_{j=2}^{J-1} \sum_{m=1}^{M_j} r_{jmk} \sum_{t=EFT_j}^{LFT_j} x_{jmt} \leqslant R_k, \quad k \in NR \tag{5}$$

$$x_{jmt} \in \{0,1\}, \quad j = 1,\ldots,J, \ m = 1,\ldots,M_j, \ t = EFT_j,\ldots,LFT_j \tag{6}$$

Constraint set (2) ensures that each activity j is performed in one of its modes and is finished within its time window $[EFT_j,\ldots,LFT_j]$. Constraints (3) represent the precedence relations. The availability per period of the renewable resource types is maintained by constraint set (4). Constraints (5) limit the total resource consumption of non-renewable resources to the available amount. The definition of all decision variables as binary is considered in constraints (6). Finally, (1) is the objective function considered in this work, to minimize the project completion time.

The feasible scheduled start/finish time (S_j/F_j) of the activities of the project is obtained as solution of this model. This model can be solved to optimality using the exact method of Talbot (1982) that was the first to present an enumeration scheme to solve the MRCPSP. Other exact approaches to solve this problem have been proposed by Patterson et al. (1989), Sprecher et al. (1997), Sprecher and Drexl (1998) and Demeulemeester et al. (2000).

Patterson et al. (1989) propose an enumerative type of branch and bound algorithm based on the generation of a precedence tree that guides the search for solutions. Sprecher et al. (1997) extend the concept of delaying alternatives introduced by Demeulemeester and Herroelen (1992) for the single-mode RCPSP, and define the concepts of delay and mode alternatives. Finally, Sprecher and Drexl (1998) present an exact procedure of the branch

and bound type in which the enumeration scheme is enhanced by search tree reduction schemes.

On the other hand, Demeulemeester et al. (2000) present a depth-first branch and bound procedure for the discrete time/resource tradeoff problem in project networks. In this work, it is considered that in real-life projects, it often occurs that only one renewable bottleneck resource is available and that the activities have a total work content which indicates how much work (expressed in man-periods) has to be performed.

However, despite the encouraging results obtained in the above mentioned exact methods, it has to be recalled that exact algorithms in general fail to solve problems with more than 20 activities thus leaving heuristics as unique alternative.

Heuristic solution procedures have been proposed by Drexl and Grünewald (1993), Özdamar and Ulusoy (1994), Boctor (1993, 1996a, b) and Kolisch and Drexl (1997). In addition, the solution procedure of Talbot (1982) as well as that devised by Patterson et al. (1989) can be applied as heuristics truncating the search procedure by imposing time limits.

Drexl and Grünewald (1993) develop models for formulating non-preemptive project scheduling problems with general resource availability and requirements as well as multimode time resource trade-offs. For the solution of this model, a stochastic scheduling method is presented which outperforms traditional deterministic scheduling rules.

Özdamar and Ulusoy (1994) present a constraint-based approach to solve the MRCPSP with $|NR| = 1$. They employ a parallel scheduling method to decide via so-called essential conditions which activity-mode pairs have to be scheduled. For each combination the increase over a lower bound of the makespan of the project is calculated and the combination that induces the smallest increase over the lower bound is then scheduled.

Kolisch and Drexl (1997) propose a local search heuristic that consists of three phases. It is interesting to note that in the computational experience they carried out neither the truncated branch and bound procedure of Talbot (1982) nor the heuristic of Drexl and Grünewald (1993) are able to find a feasible solution to all the project instances considered. However, the heuristic of Kolisch and Drexl (1997) is able to find a feasible project schedule for all the project instances considered.

On the other hand, Boctor (1993, 1996a, b) describes heuristic solution methods for the MRCPSP when only renewable resource types are considered. In the first work, the author proposes several heuristics based on priority rule with a modified version of the parallel schedule generation scheme and several rules are used both to sort the activities in the eligible set and to select the mode of the activities. Among the best heuristics are the ones based on the Minimum Latest Finish Time and the Minimum Slack rules calculated assigning to each activity its mode with the shortest duration. In Boctor (1996a) a deterministic heuristic that enumerates some schedulable combinations of activities and chooses from them the one having the best value for an evaluation criterion is proposed. Finally, in Boctor (1996b) a simulated annealing algorithm is presented. The heuristic is able to handle single-mode and multimode problems and to consider different objective functions.

Recently, efforts have been addressed towards the application of metaheuristics such as those by Özdamar (1999), Hartmann (2001), Józefowska et al. (2001), Alcaraz et al. (2003), Bouleimen and Lecocq (2003) for the MRCPSP when renewable and non-renewable resource constraints exist. Özdamar (1999) develops a GA based on an encoding which is made up by a sequence of priority rules and a mode assignment. As decoding procedure, parallel schedule generation scheme is used. For each individual two schedules are computed by scheduling forward and backward. Hartmann (2001) and Alcaraz et al. (2003) apply GAs that use precedence activity list (AL) and a mode assignment list as representation of one individual. In addition, Alcaraz et al. (2003) extend the representation given for RCPSP by adding an extra gene for backward or forward scheduling direction. They use different fitness function and both of them apply a simple local search procedure to non-feasible mode assignments in the initial population. On the other hand, Józefowska et al. (2001) and Bouleimen and Lecocq (2003) use simulated annealing algorithms. Concretely, Bouleimen and Lecocq (2003) propose a new design of the conventional simulated annealing search scheme taking into account the specificity of the solution space of the project scheduling problems.

3. Description of the new hybrid GA

Before the GA itself is started, a pre-processing procedure over the project data in order to reduce the search space is applied. This procedure was introduced by Sprecher et al. (1997) to reduce the amount of data and speed up the execution of their algorithm. This procedure has been used later on by Hartmann (2001) and Alcaraz et al. (2003) in their GAs. The reduction procedure consists of excluding inefficient modes, non-executable modes and non-renewable resources from the input data. Following Sprecher et al. (1997) procedure, within a feasible schedule no activity can be performed in a non-executable mode. If there is an optimal schedule for a given instance, then there is an optimal schedule in which no activity is accomplished in an inefficient mode. Finally, excluding a redundant non-renewable resource from a project instance does not affect the set of the feasible (optimal) schedules. A formal definition of inefficient modes, non-executable modes and redundant non-renewable resources can be found in Sprecher et al. (1997).

After the application of the pre-processing procedure, the execution of the GA itself starts. First, the initial population (P) or first generation is generated and each individual is transformed into a schedule and evaluated. Once all the individuals in the initial population have been assigned a fitness value, the following steps are repeated until the terminating condition (execution time or number of feasible solutions or number of generations) is reached. First a selection mechanism makes the best individuals to have a higher probability of surviving for

```
MM-HGA  (POP_SIZE, end_cond)
begin
P = Generate_Initial_Population(POP_SIZE)
While NOT (end_cond) do
begin
P = Selection(P)
P = Crossover(P)
        P = Mutation(P)
        For each j in P
             If j is feasible with respect to NR
                  j = MM-FBI(j)
        BEST_Individual = Evaluate_Population(P)
end
return BEST_Individual
end
```

Fig. 1. Hybrid Genetic Algorithm (MM-HGA).

the next generation. The population is randomly divided into pairs of individuals and each pair undergoes the crossover procedure to produce offspring with a given probability. Then, a mutation procedure is applied to modify some of the individuals before a new population is obtained. Finally, an improving procedure is applied to each individual whenever it is feasible w.r.t. non-renewable resources. Fig. 1 summarizes this procedure (MM-HGA) each iteration of which corresponds to a new generation of individuals. The main characteristics of each step of the algorithm are detailed in the following sections.

3.1. Solution encoding and decodification process

In order to apply a GA to a particular problem, an internal representation for the solution space is needed. The choice of this component is one of the critical aspects for the success/failure of the GA for the problem under study. Kolisch and Hartmann (1999) distinguished five different schedule representations in the RCPSP literature, from which the AL representation and the random-key (RK) representation are the most widespread. In both representations, a priority structure between the activities is embedded. In the AL representation, the position of an activity in the AL determines the relative priority of that activity regarding the other activities, while in the RK representation the position of an activity is based on the priority value attributed to an activity. Hartmann and Kolisch (2000) concluded from experimental tests that procedures based on AL representations outperform the other procedures.

In this work, we have used an AL as representation of a solution. Each individual is represented by a double list: an AL and a mode assignment list both with as many positions as activities in the project. In the AL each activity is placed after all its predecessors, therefore it is precedence feasible. The mode assignment represents the execution mode of each activity. This representation has been used by several authors such as Hartmann (2001), Józefowska et al. (2001) and Bouleimen and Lecocq (2003) amongst others. Alcaraz et al. (2003) add to this representation an additional gene forward/backward (f/b) that indicates the direction in which the serial scheduling generation scheme is used to build the schedule.

Indeed, the serial schedule generation scheme has been the most widely used as a decoding procedure

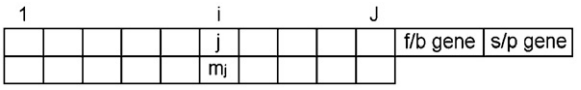

Fig. 2. Representation of an individual.

mainly because it is immediately applied to AL representations. However, the consideration of both schedule generation schemes would benefit the solving process by allowing to explore different regions of the search space. Thus, we propose to add an additional gene to the encoding of each individual called serial/parallel (s/p) indicating the Scheduling Generation Scheme (SGS) used to generate the schedule. The Serial SGS can be applied directly from this representation to transform the individual into its corresponding schedule; on the other hand the P-SGS uses the position of the activity as a priority rule to select the activity to be scheduled. The codification proposed for the MRCPSP is illustrated in Fig. 2.

3.2. Fitness computation

When applying the GA, we need to define an evaluation function that determines the probability of survival of an individual to the next generation. Frequently the fitness function coincides with the objective function because it is an accurate measure of how good the individual is. In MRCPSP when the objective is to minimize the project duration, the makespan of the schedule related to an individual should be a good fitness value unless for the fact that some individuals of each population can be infeasible with respect to non-renewable resources. These individuals must also be assigned a value indicating their fitness. Furthermore, the fitness value of infeasible individuals should not be better than that of feasible ones, that is to say infeasible individuals have to be penalised but considered in the selection process. Different evaluation or fitness functions have been proposed in the literature taking into account the just mentioned considerations.

Hartmann (2001) defined the following fitness function (also used later by Józefowska et al., 2001):

$$f(I) = \begin{cases} mak(I) & \text{if } SFT(I) = 0 \\ T + SFT(I) & otherwise \end{cases} \quad (7)$$

where $mak(I)$is the makespan of the individual I and T is the upper bound on the project's makespan that is given by the sum of the maximal durations of the activities. $SFT(I)$is the excess of non-renewable resources defined by

$$SFT(I) = \sum_{k \in NR} \max\left\{0, \sum_{j=1}^{J} (r_{jmk} - R_k)\right\} \quad (8)$$

Therefore an individual is a feasible schedule *if and only if* $SFT(I) = 0$.

Alcaraz et al. (2003) stated the following weak points of this fitness function:

(i) The fitness value of an infeasible individual does not consider its duration and only depends on the excess of non-renewable resources.

(ii) The upper bound T is so poor that an infeasible individual will have a probability close to zero and will not participate in the genetic process.

To overcome these drawbacks Alcaraz et al. (2003) propose the following fitness function:

$$f(I)=\begin{cases} mak(I) & \text{if } SFT(I)=0 \\ \max\_mak(P)+mak(I) & \\ \quad -\min\_CP+SFT(I) & \text{otherwise} \end{cases} \tag{9}$$

where $mak(I)$ is the makespan of the individual I and $\max\_mak(P)$ gives the maximal makespan of feasible solutions related to individuals of the current generation which represents an upper bound of the project's makespan. To this bound is added the excess of non-renewable resources $SFT(I)$ and the increase over the makespan given by the minimal critical path, $\min\_CP$ using the minimal duration of activities.

With this fitness function two solutions with identical SFT but different makespan will have a different fitness value and the penalty of a non-feasible individual gives reasonable probabilities to participate in the genetic process.

However, this fitness function is built by adding units of time from the makespan and units of resources from the excess of non-renewable resources. The magnitude of both aspects of the solution can disturb the meaning of the fitness function. To solve this weak point, we propose a new fitness function where both aspects of the solution are jointly considered but normalized in order to eliminate their magnitudes.

Therefore, we propose a new fitness function that is computed for each individual according to the following expressions depending on whether individual I is feasible ($SFT(I)=0$) or non-feasible:

$$f(I)=\begin{cases} 1-\dfrac{\max\_mak(P)-mak(I)}{\max\_mak(P)} & \text{if } SFT(I)=0 \\ 1+\dfrac{mak(I)-\min\_CP}{mak(I)} & \\ \quad +\displaystyle\sum_{k\in NR}^{K}\max\left\{0,\frac{\sum_{j=1}^{J}r_{jmk}-Rk}{Rk}\right\} & \text{otherwise} \end{cases} \tag{10}$$

The feasible individual with the greatest makespan will have a fitness value equal to 1 while the best one will have a fitness value close to zero.

The fitness function value of a non-feasible individual always is greater than 1. Then, all non-feasible individuals will have a fitness value greater than that of the feasible ones (and will have fewer opportunities for survival in the selection process). In addition, the sum of the normalized deviation of the makespan from the minimal critical path and the normalized excess of non-renewable resources are added. This fitness function solves weak points of fitness computations for the MRCPSP that appeared in the literature.

In order to test the performance of this fitness function against the ones proposed by Hartmann (2001) and Alcaraz et al. (2003) with the same computational effort, the well-known PSPLIB multimode sets of instances[1] J10, J12, J14, J16, J18 and J20 have been solved by the hybrid GA developed in this work and the three fitness functions under consideration. Results are reported in Table 1 and show that the fitness function proposed in this work has a more robust behaviour thus making more appropriate its use in the MRCPSP. Concretely, in 4 out of the 6 instance sets the new fitness function obtains the best results (measure as the deviation from the optimum). Furthermore we can show by a t-test over all instance that the new fitness function gives better results than the other ones with a 90% level of confidence.

Table 1
Performance of the fitness functions: (a) fitness function comparison (average deviation with respect to the optimal solution (%)), (b) percentage of optimally solved instances.

Fitness function	J10	J12	J14	J16	J18	J20
(a)						
This paper	0.06	0.17	0.32	0.44	0.63	0.87
Alcaraz et al. (2003)	0.07	0.16	0.33	0.48	0.67	0.93
Hartmann (2001)	0.10	0.18	0.35	0.51	0.65	0.85
(b)						
This paper	98.51	96.53	92.92	90.00	84.96	80.32
Alcaraz et al. (2003)	98.32	96.71	92.20	89.09	83.88	79.24
Hartmann (2001)	97.76	96.34	91.83	88.36	84.24	80.87

3.3. Initial population

The GA starts with the generation of the initial population, that is, a set of POP_SIZE solutions. Each individual is obtained randomly selecting the genes p/s and f/b and a selection mode described below. With this information, each solution is obtained using the heuristic based on the Minimum Latest Finish Time (LFT) priority rule.

3.3.1. Activity selection mode

The MRCPSP in reference to RCPSP poses an additional problem since not all mode assignments are feasible with respect to the non-renewable resources. For this purpose we propose a procedure to maximize the probability of obtaining feasible solutions in the initial population thus starting the evolution process of the GA with a good set of solutions.

The probability of obtaining a feasible mode assignment following this procedure is very high and it does not require any additional computational effort. This procedure that we have called Minimum Normalized Resources (MNR) implies the selection for each activity j of the mode with minimum value NR:

$$NR_{jm}=\sum_{k\in NR}\frac{r_{jmk}}{R_k} \tag{11}$$

[1] More details about instances are given in the Computational experience section.

To illustrate the application of this criterion Table 2 shows the different execution modes of an activity that uses two non-renewable resources R_1 and R_2. Their total availabilities are $R_1 = 40$ and $R_2 = 20$, respectively. Following the MNR criterion calculated in the forth column, the second mode will be chosen.

Table 3 shows the percentage of feasible mode assignments generated by the MNR mode selection method, random selection mode (Random) and the selection of the mode with longest duration (MaxDur) for the set of instances J10, J12, J14, J16, J18, J20 and J30 of PSPLIB (Kolisch and Sprecher, 1996).

The MNR selection mode procedure obtains feasible non-renewable assignments in a range that varies from 95.61% to 98.73% of the cases, while the MaxDur selection mode (that usually selects the mode with the lowest requirement of resources) obtains feasible mode assignments in a range that varies between 90.30% and 95.29% of the cases. Finally, Random selection mode assigns feasible modes only in a short number of cases.

The first individual of the initial population is generated by the MNR procedure that obtains a feasible non-renewable assignment with a very high probability In order to add diversity, the mode assignment of the remaining individuals of the initial population is based on a randomized version of the MNR criterion. For each individual the process is the following:

Step 1. The process starts with the MNR mode assignment.
Step 2. The mode of at most $J/2$ activities is randomly changed.
Step 3. If the resulting mode assignment is feasible, *then* STOP.
Otherwise, activities are randomly sorted and each activity tries to change its current mode to another one able to reduce the level of infeasibility of the non-renewable resources (if it exists).
Step 4. If the resulting mode assignment is feasible, *then* STOP.
Otherwise, Go to *Step* 1

Table 2
MNR example.

Mode	r_{jm1}	r_{jm2}	MNR
1	0	7	$0/40 + 7/20 = 0.35$
2	3	0	$3/40 + 0/20 = 0.075$
3	0	2	$0/40 + 2/20 = 0.10$

This process is repeated up to a pre-established number of attempts (*Att*) that in this paper is set to 200.

At first glance it can be thought that this procedure of obtaining a feasible non-renewable mode assignment is very time consuming. However, in practice a feasible non-renewable mode assignment is obtained in a percentage of individuals varying from 79.71% to 86.69% and in more than 96% of the individuals, a maximum of 3 attempts are needed. Data corresponding to the different sets of instances are reported in Table 4.

3.4. Crossover operator

One of the unique and important aspects of the techniques involving GAs is the important role that recombination (traditionally, in the form of crossover operator) plays. Crossover combines the features of two parent chromosomes to form two offspring that inherit their characteristics. The individuals of the population are mated randomly and each pair undergoes the crossover operation with a probability of P_{cross}, producing two children by crossover. The parent population is replaced by the offspring population. The crossover is one of the most important genetic operators and must be correctly designed. Crossover must combine solutions to produce new ones. Crossover must preserve and combine "good building blocks" to build better individuals Given two individuals selected for crossover, a mother **M** and a father **F**, two offspring, a daughter **D** and a son **S** are produced.

We have implemented the well-known 2-point *crossover* with $P_{\text{cross}} = 0.9$. Firstly, two integer and non-negative crossover points k_1 and k_2 are randomly generated ($k_2 > k_1$). The offspring ALs and mode assignment lists are generated as follows. The first k_1 positions in the Son (S) and those from $k_2 + 1$ to J are inherited from the father, exactly in the same order. The positions

Table 4
Percentage of individuals with feasible non-renewable mode assignments in the initial population according to the number of attempts needed.

Attempts	J10	J12	J14	J16	J18	J20	J30
0[a]	79.71	82.95	82.64	83.84	85.03	84.61	86.69
1	11.90	10.59	10.00	9.54	8.47	8.38	6.86
2	3.55	3.00	2.99	2.72	2.57	2.52	2.21
3	1.69	1.20	1.27	1.32	1.34	1.36	1.35
4–199	3.14	1.92	2.67	2.13	2.16	2.57	2.57
200	0.00	0.33	0.43	0.45	0.42	0.55	0.32

[a] A feasible mode assignment is obtained in Step 3.

Table 3
Percentage of feasible mode assignments.

Method of selection mode	J10	J12	J14	J16	J18	J20	J30
Random	53.92	56.12	55.17	54.91	55.80	58.12	59.78
MaxDur	90.30	93.42	91.47	93.27	94.38	94.04	95.29
MNR	96.08	95.61	97.46	97.27	98.73	97.83	98.55

between k_1+1 and k_2 are inherited from the mother's sequence preserving their relative order. More precisely, let i be the ith position in the list S, then S directly inherits the positions of the father from the interval $i=1$ to k_1 and from interval $i=k_2+1$ to $i=J$. Finally, positions from $i=k_1+1$ to $i=k_2$ of S are inherited from the mother. Mother's sequence is analyzed from 1 to J and the position is inherited by the Son whenever it was not included in the interval $[1,\ldots,k_1]\cup[k_2+1,\ldots,J]$ of S. The mode list of S is generated with the mode of the corresponding activities inherited from the father and the mother. In this way, the solution generated, the Son, is a precedence feasible solution. The s/p and the b/f genes are inherited from the first progenitor. The Daughter (D) is generated analogously. Fig. 3 illustrates this procedure assuming $k_1=2$ and $k_2=7$ and using the project instance of Table 5.

3.5. Mutation operator

Once the crossover operator has been applied and the offspring population has replaced the parent population, the mutation operator is applied to the offspring population. Mutation alters one or more genes (positions) of a selected chromosome (solution) to reintroduce lost genetic material and introduce some extra variability into the population. In fact, mutation can result in entirely new gene values that sometimes allow the GA to reach solutions better than those previously possible. In addition, mutation helps to prevent the population from stagnating at any local optima.

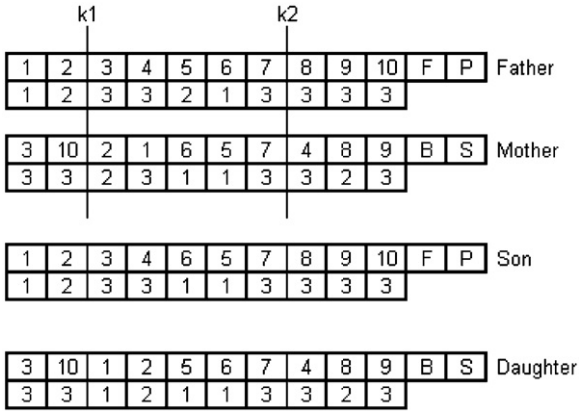

Fig. 3. Crossover operator.

Mutation is applied to both components of each individual representation that are AL and mode assignment list. In the first case, the mutation procedure used is the insertion operator that works as follows: for each activity in the AL a new position is randomly chosen between the highest position of its predecessors and the lowest position of its successors. The activity is inserted into the new position with a probability of P_{mut}. In the GA developed, P_{mut} equals 0.05.

Concerning the mode assignment list, the application of the mutation operator changes depending on whether the individual has or not a feasible non-renewable mode assignment:

1. If the individual has a non-renewable feasible mode assignment each activity changes randomly its mode with the same probability P_{mut}.
2. If the individual mode assignment is non-feasible w.r.t. non-renewable resources, a new mutation operator, called *massive mutation*, is applied with the objective of increasing diversity of the mode assignments. This operator works as follows: for each activity randomly chosen from AL a new mode is randomly assigned (that could be the same as before). This process ends either when the current non-renewable mode assignment is feasible or when all activities of the AL have been considered.

Finally, genes s/p and b/f change with a probability P_{mut}.

Using the project instance of Table 5, Fig. 4 illustrates the mutation procedure used. Assuming activity 6 randomly chosen for mutation and that its latest predecessor is activity 3 and its earliest successor is activity 7, a random number from 4 to 6 (the position of the activities) has to be generated. The value obtained equals 3 and therefore activity 6 is inserted in that position (Fig. 4b).

Table 5
Project instance—availability: (R1, R2, NR3, NR4) = (12, 12, 37, 60).

Activity	Immediate successor activities	d_{j1}	Mode 1				d_{j2}	Mode 2				d_{j3}	Mode 3			
			R		NR			R		NR			R		NR	
			r_{j11}	r_{j12}	r_{j13}	r_{j14}		r_{j21}	r_{j22}	r_{j23}	r_{j24}		r_{j31}	r_{j32}	r_{j33}	r_{j34}
1	4, 5, 10	2	5	9	4	7	5	5	8	2	7	6	5	6	1	6
2	6	1	6	5	6	8	8	5	4	5	5	8	4	5	4	6
3	6, 10	1	4	9	9	5	8	3	4	6	1	8	1	4	2	3
4	8, 9	1	7	3	7	8	2	6	3	5	8	2	5	3	6	5
5	7	1	7	9	4	10	4	7	7	2	10	9	6	2	1	10
6	7, 8	1	9	5	3	6	1	9	4	3	7	5	9	3	3	6
7	9	2	9	5	9	8	2	9	6	7	8	3	9	4	4	8
8	–	1	9	9	10	6	5	9	8	6	6	9	9	8	4	5
9	–	2	7	2	4	9	9	7	2	4	6	10	3	1	3	3
10	–	3	5	10	8	10	6	4	10	5	7	7	4	9	3	7

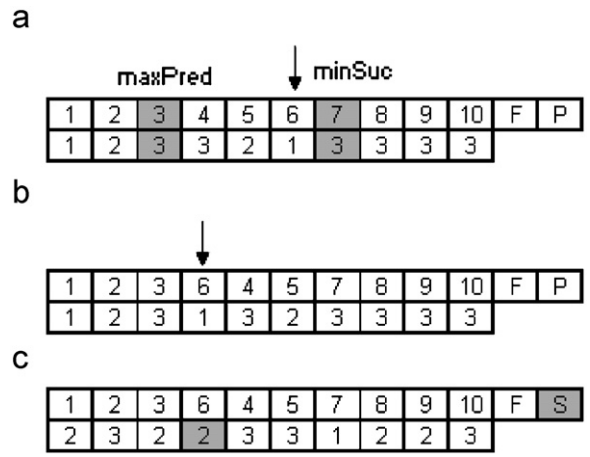

Fig. 4. Activity insertion mutation operator.

Since the non-renewable mode assignment is feasible, the mode of the activity is mutated with the same probability and in this case changes to mode 2 (Fig. 4c). The f/b gene remains unchanged and the p/s gene varies to *s*.

3.6. Selection operator

Selection is an artificial version of the natural phenomenon called the survival of the fittest. In nature, competition among individuals for scarce resources and for mates results in the fittest individuals dominating over weaker ones. Based on their relative quality or rank, individuals receive a number of copies. A fitter individual receives a higher number of offspring and, therefore, has a higher probability of surviving in the subsequent generation. There are several ways of implementing the selection mechanism.

We have implemented *2-tournament* selection. This selection mechanism implies that two individuals are randomly chosen from the population and compete for survival. The best of them, i.e. the one with the best fitness value will appear in the subsequent population. This procedure is repeated POP_SIZE times, until POP_SIZE individuals are selected to appear in the next population. Elitism is applied thus guaranteeing the survival of the best individual.

3.7. Replacement of the population

In order to design a good GA, a random replacement procedure of the population is necessary. This operation is applied to each generation with a given probability of P_{replac}. If it is performed in the current generation, each individual of the population is exchanged with a probability of P_{exchange} by a solution randomly generated using the MNR procedure described in Section 3.3 with $Att = 1$. This procedure allows restarting the population or to reintroduce some variability when it has prematurely converged or it is stuck in a local optimum. The values of P_{replac} and P_{exchange} are set to 0.7 and 0.1 respectively.

3.8. The multimode forward–backward improvement method (MM-FBI)

The project completion time of each feasible schedule generated by the GA can be reduced by means of the following local search technique. The proposed method is an extension of the FBI method described by Tormos and Lova (2001) (also called *justification*, Valls et al., 2005) and widely used in recent papers to approximately solve the RCPSP. The consideration of multiple modes for activities requires the extension of the method that in addition to the change of the scheduling time usually implies the change of the execution mode looking for a new optimized position of each activity. The resulting method is applied to each feasible individual drastically reducing its duration.

A feasible schedule for the MRCPSP consists of a mode assignment M and a schedule S, denoted as (M,S). The mode assignment M is a J-tuple $M = (m(1),\ldots,m(J))$ that assigns to every activity j a unique mode $m(j)$. The schedule S is a J-tuple $S = (S(1),\ldots,S(J))$ that assigns to every activity j a unique start time Sj.

The feasible project schedule can be improved, i.e. the project completion time reduced, with the application of a multimode backward–forward method (MM-BF) or the application of a multimode forward–backward method (MM-FB) depending on whether the initial feasible schedule is obtained with the gene f/b set to f or b, respectively. Each iteration of these methods consists of a Multimode Backward pass and a Multimode Forward pass. The implementation of the MM-BF and the MM-FB methods is based on ALs that works on what is called the topological order of the graph, that is, a precedence ordered AL where each activity in the list has all its successors/predecessors in earlier positions. Another important difference between the improving method proposed in Tormos and Lova (2001) and its extension to the MRCPSP presented in this work is that while in the first improving method, the MM-BF and MM-FB methods are based on the completion of partial feasible schedules, in the currently adapted method, a feasible schedule of the project always exist, and a rescheduling process is applied in order to reduce the project completion time. That is to say, when an activity changes its feasible start/finish time, the remaining activities have a feasible start/finish time.

The *multimode backward pass* (MM-B) takes each activity i in non-increasing order of its scheduled finish time. The activity i is rescheduled in the feasible mode (renewable and non-renewable resources) that allows the activity to be scheduled at the latest feasible start time in the time window delimited by $[S_i, \min(S'_j - d_{im} | m$ belongs to $M_i) \forall \text{Precedence}(i,j)]$, S'_j being the scheduled start time of activity j in the updated feasible schedule. If ties occur the activity mode with the shortest duration is chosen. Thus, this process can imply a change of the mode assignment of activity i in order to expand the time window of the predecessor activities.

Once all activities in the AL have been evaluated, if $S^{\min} = \min(S_j)$ is greater than zero it implies to reduce the completion time of the project by $S^{\min}$ units of time. In such a case the scheduled start time of each activity is

updated ($S_j = S_j - S^{min}$) as well as their scheduled finish time according to $F_j = F_j - S^{min}$.

The *multimode forward pass* processes (MM-F) each activity i of the feasible schedule in non-decreasing order of its scheduled start time. Each activity i is rescheduled in the feasible mode assignment (renewable and non-renewable resources) that allows the activity to be scheduled at its earliest feasible finish time in the time window delimited by $[\max(F'_j)\forall \text{Precedence}(i,j), S_i]$, F'_j being the finish time of activity j in the updated feasible schedule. If ties occur the activity mode with the shortest duration is chosen. Thus, this process can imply a change of the mode assignment of activity i addressed to expand the time window of the successor activities.

It is important to note that when an activity is rescheduled both in MM-F and MM-B only those modes that maintain the non-renewable feasibility are considered. Thus, feasibility with respect to the non-renewable resources is always preserved even if the mode assignment were changed in forward and backward passes.

The MM-BF method starts from the feasible schedule obtained with forward scheduling direction (f gene) and each iteration implies the application of the Multimode Forward pass after the application of the Multimode Backward pass. On the other hand, the MM-FB method starts from the feasible schedule obtained with backward scheduling direction (b gene) and each iteration implies the application of the Multimode Backward pass after the application of the Multimode Forward pass. Hence, each MM-FBI pass (MM-FB or MM-BF pass) obtains one feasible schedule with a project completion time lower than or equal to the previous one. Since additional iterations can lead to further reducing the project completion time, MM-FB or MM-BF is applied until no improvement is achieved.

This extension of the Backward–Forward method applied to the RCPSP by Tormos and Lova (2001) to the MRCPSP case (MM-FBI) is an effective technique to reduce the feasible project completion time when renewable and/or non-renewable resources constraints exist.

3.8.1. A numerical example

In order to illustrate the multimode improving technique proposed in this work, we will apply it to a project instance with 10 non-dummy activities that require two of two constrained renewable resources and two of two non-renewable resources. Concretely, the project instance used is j1037_2, belonging to the J10 set of instance generated by Kolisch and Sprecher (1996). This project is one of the most difficult to solve since all activities require units of two highly constrained renewable resources and units of two highly constrained non-renewable resources. In Table 5 is shown for each activity, its immediate successor activities, the duration and the resource requirements for each mode.

Using the feasible mode and the activity start time shown in Table 6 (second column), an initial feasible project solution is obtained with forward scheduling direction (Fig. 5a). The project duration is 38 periods of time and this feasible schedule becomes the input data for the improving method MM-FBI.

Table 6
Project instance: mode assignment and start time in each stage of the new improving method.

Activity	Initial feasible solution		MM-BFI			
			MM-B		MM-F	
	Mode assignment	S_j	Mode assignment	S_j	Mode assignment	S_j
1	1	0	1	0	1	0
2	2	2	2	3	2	2
3	3	6	3	3	3	2
4	3	6	3	19	3	15
5	2	2	3	2	3	2
6	3	14	1	11	1	11
7	3	19	3	12	3	12
8	3	29	3	22	2	22
9	3	22	3	21	3	17
10	3	22	3	15	3	15

After the application of the MM-B pass, the project completion time is reduced by 7 periods of time (Fig. 5b). The MM-B implies also the change of the mode assignment of activities 5 and 6. Their final mode assignments are also shown in Table 5.

When the MM-F pass is applied to the feasible schedule just obtained by the MM-B pass, the project completion time is reduced again by 4 periods of time (Fig. 5c). Activity 8 changes its mode assignment and the final activity mode assignment is shown in Table 6.

The final solution obtained after applying one MM-FBI iteration has duration of 27 periods of time that means a makespan 11 time units shorter than the initial feasible schedule. This duration corresponds to the optimal solution for the instance considered. In other projects, additional MM-BF iterations can be needed to further reduce the project completion time.

4. Computational experience

The well-known sets of instances generated by the project generator ProGen for the MRCPSP and available at http://129.187.106.231/psplib/ have been used to test the performance of the hybrid GA proposed. Concretely these sets for the multimode problem contain instances with 10, 12, 14, 16, 18, 20 and 30 non-dummy activities. Each of the non-dummy activities may be performed in one out of three modes that can require units of two renewable and two non-renewable resources. The duration of a mode varies between 1 and 10 periods of time. The set of instances with 20 non-dummy activities currently is the hardest standard set of multimode instances for which all optimal solutions are known. Consequently, the quality of a heuristic is usually measured in terms of deviation with respect to the optimum. However, for the set of instances with 30 activities not all optimal solutions are known, then the measurement is based on the deviation with respect to the critical path when the shortest mode for each activity is used. The computational experiments have been carried out on a PC Pentium with 3 GHz and 1 GB

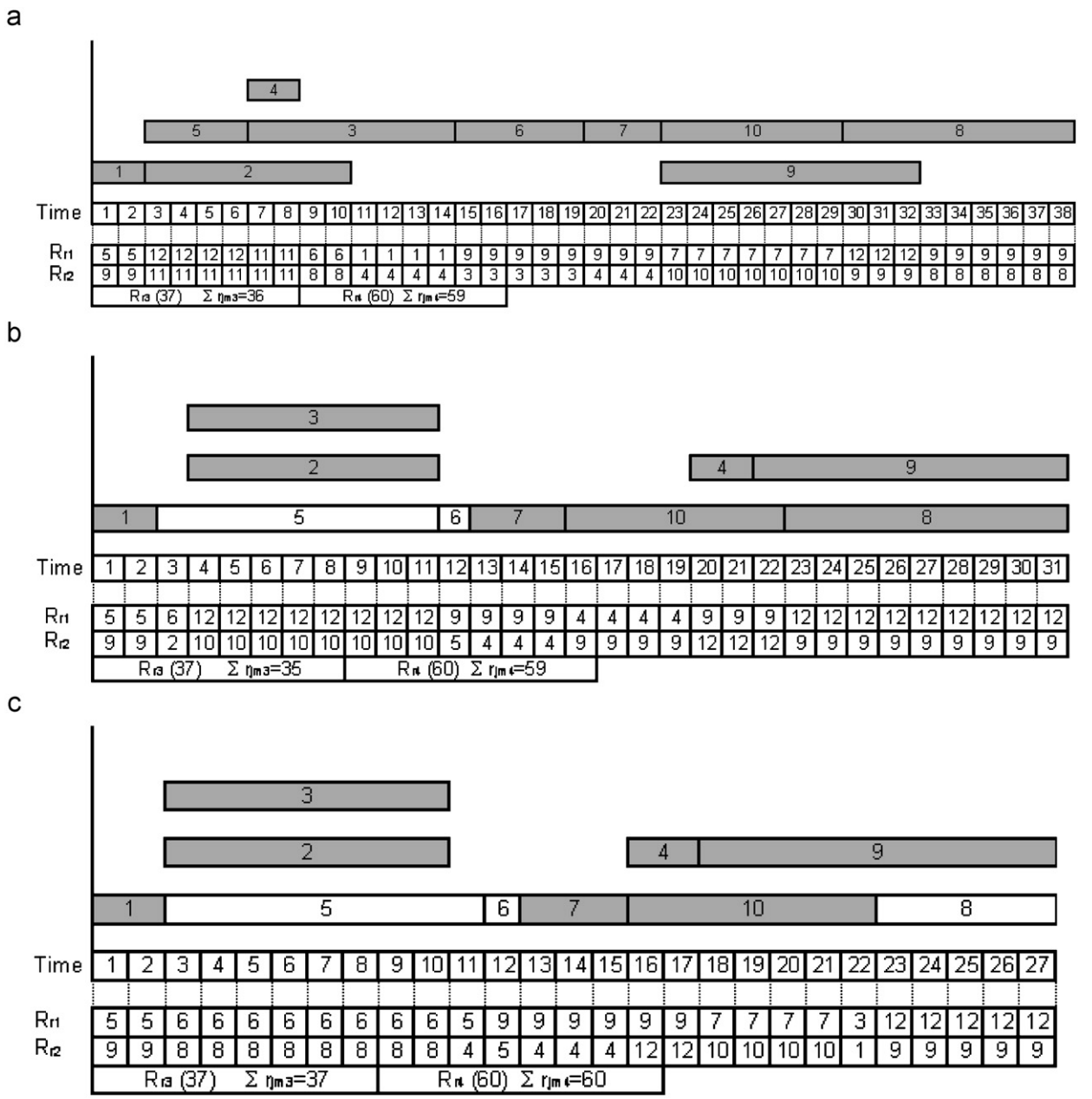

Fig. 5. (a) Initial feasible solution, (b) solution after applying MM-B, (c) solution after applying MM-F.

RAM using an implementation in C compiled with Microsoft Visual C++ v.6.0 under Windows XP.

When comparing the performance of a set of algorithms it is indispensable to make the comparison on the basis of the same computational effort regardless the computer used. With this idea in mind since the work by Hartmann and Kolisch (2000) the performance of different algorithms is compared on the basis of the number of schedules generated to achieve a given result or solution. Furthermore, more recently Hartmann and Kolisch (2006) have specified this approach even more and they propose that the computational effort to generate one schedule should correspond to (at most) one start time assignment per activity, as done by an SGS. Obviously, the computational effort of one MM-FBI pass is greater than the one needed to generate two SGSs. For this reason, in this work the computational effort to generate one MM-B or one MM-F equals the sum of times each activity of the project has obtained a feasible start time divided by the number of activities of the project. Furthermore, regardless of whether the start and finish time of each activity changes or not, at least one feasible start time is obtained and computed for each activity. For instance, in the project used in Section 3.8.1, the application of the MM-B and the

MM-F is equivalent to the computational effort when 1.1 and 1.6 SGSs are generated, respectively.

4.1. Performance of the massive mutation operator in the Simple GA

The first analysis that we have performed is the study of the influence of the massive mutation operator on the Simple GA (i.e. the MM-FBI method is not applied).

Fig. 6 illustrates the results for both J10 and J20 sets of instances when the number of schedules varies from 500 to 6000. In all cases, the application of the massive mutation leads to better results also regardless the number of schedules generated. As conclusion, the massive mutation is included in the MM-HGA (Table 7).

4.2. Performance of the incorporation of the MM-FBI method to the Simple GA

The influence of the MM-FBI on the GA has been evaluated by solving the J10 and J20 sets of instances with a computational effort that varies from 500 to 6000 schedules. Fig. 7 illustrates the results with and without using the local search process. The application of the MM-FBI proves to be decisive by drastically improving the performance of the GA. When the J10 set is considered, the GA without the inclusion of the MM-FBI gives a deviation with respect to the optimal solution that varies from 3.27% to 0.16% depending on the computational effort. However, when the MM-FBI is included the deviation w.r.t. the optimal solution drops until values that vary from 0.59% to 0.04%. Conclusions are similar

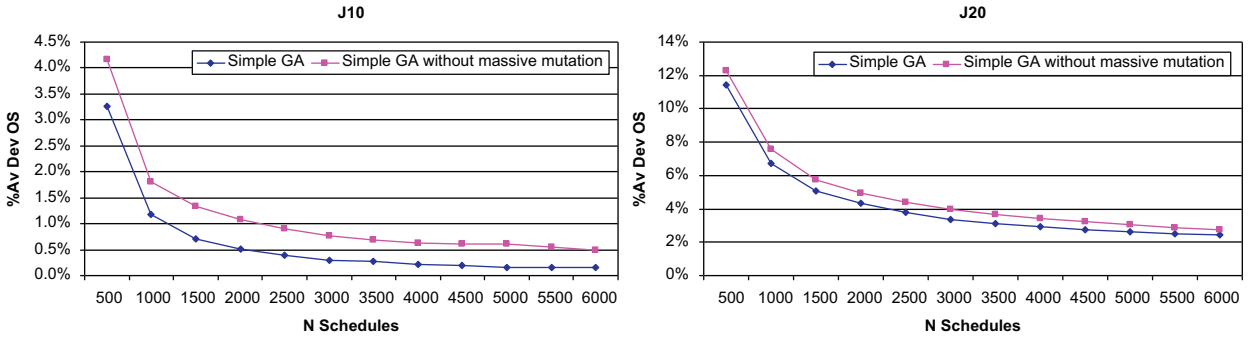

Fig. 6. Impact of the massive mutation operator on the basic GA.

Table 7
Impact of the massive mutation operator on the basic GA.

#Sched	J10				J20			
	Simple GA without massive mutation		Simple GA		Simple GA without massive mutation		Simple GA	
	Av.Dev.OS (%)	CPU-time (s)	Av.Dev.OS (%)	CPU-time (s)	Av.Dev.OS (%)	CPU-time (s)	Av.Dev.OS (%)	CPU-time (s)
500	4.16	0.01	3.75	0.01	12.26	0.02	12.39	0.02
1000	1.81	0.02	1.31	0.02	7.56	0.03	6.99	0.04
2000	1.07	0.03	0.50	0.03	4.94	0.06	4.41	0.06
3000	0.77	0.05	0.30	0.05	3.95	0.09	3.46	0.09
4000	0.63	0.06	0.22	0.06	3.44	0.12	3.00	0.12
5000	0.61	0.08	0.16	0.08	3.07	0.15	2.61	0.15
6000	0.49	0.09	0.16	0.09	2.77	0.17	2.51	0.17

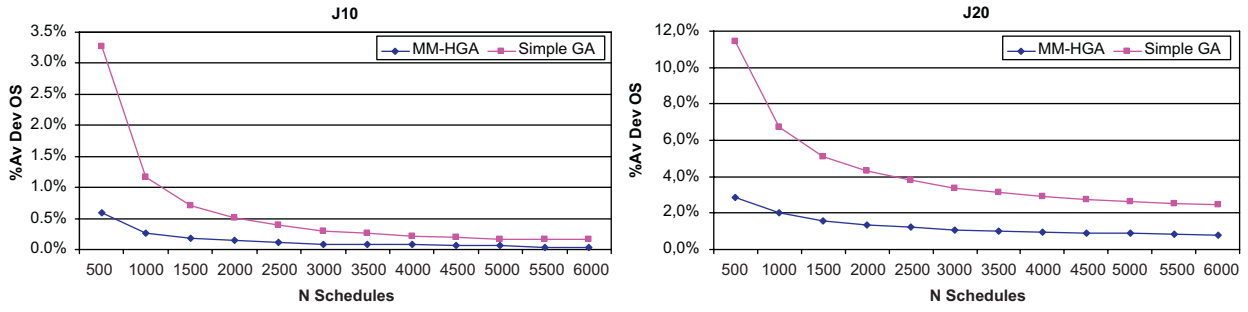

Fig. 7. Performance of the Simple GA and MM-HGA with the J10 and J20 sets of instances.

Table 8
Performance of the Simple GA and MM-HGA with the J10 and J20 sets of instances with a computational effort that varies from 500 to 6000 schedules generated.

#Sched	J10				J20			
	Simple GA		MM-HGA		Simple GA		MM-HGA	
	Av.Dev.OS (%)	CPU-time (s)	Av.Dev.OS (%)	CPU-time (s)	Av.Dev.OS (%)	CPU-time (s)	Av.Dev.OS (%)	CPU-time (s)
500	3.75	0.01	0.59	0.01	12.39	0.02	2.90	0.02
1000	1.31	0.02	0.26	0.02	6.99	0.04	2.08	0.04
2000	0.50	0.03	0.14	0.03	4.41	0.06	1.38	0.06
3000	0.30	0.05	0.09	0.05	3.46	0.09	1.08	0.08
4000	0.22	0.06	0.08	0.06	3.00	0.12	0.98	0.12
5000	0.16	0.08	0.06	0.08	2.61	0.15	0.87	0.14
6000	0.16	0.09	0.04	0.09	2.51	0.17	0.82	0.18

when the J20 set of instances is considered (Fig. 7 and Table 8).

4.3. Performance of the new MM-HGA for the MRCPSP

In order to test the performance of the hybrid GA proposed, the J10, J12, J14, J16, J18, J20 and J30 PSPLIB set of instances are used. Each set of instances is solved with the computational effort of 1000, 3000 and 5000 schedules generated. Table 8 shows for each experiment the average deviation with respect to the optimal solution/critical path based lower bound (Av.Dev.OS/Av.Dev.LB), the maximum (Max.) and the standard deviation (St.Dev.), the percentage of optimum solutions found (%Optimum) and the CPU time required in seconds.

Instances used in the evaluation reported in Table 9 can be considered as small-sized projects. Therefore in order to evaluate the performance of the MM-HGA when solving medium- and large-sized project instances we have designed two additional sets of instances J60 and J120. Both sets of instances have been generated using the instance generator ProGen and have similar characteristics as those of previously existing sets. From these results is concluded the good behaviour of the algorithm even when the project size increases (Table 10).

4.4. Performance of the new hybrid GA against the best metaheuristic algorithms published

In order to compare the performance of the new HGA with that of the best algorithms published so far, we have solved the multimode J10 set of instances with the MM-HGA developed in this work considering a computational effort of 1000, 2000, 3000, 4000, 5000 and 6000 schedules generated. Results are drawn in Table 11 where the average deviation with respect to the optimal solution, the percentage of optimal solutions found, the computational effort (measured as the number of schedules generated) and the mean CPU-time in seconds are reported. We can see that the hybrid GA proposed (MM-HGA) outperforms the heuristic and metaheuristic methods previously published in the literature when the same computational effort of 6000 schedules generated is considered. The MM-HGA obtains an average deviation with respect to the Optimum of 0.04% while the next best algorithm obtains a 0.1%. It is interesting to point out that MM-HGA outperforms the best algorithm published so far even with half of the computational effort. The algorithm maintains the rate of improvement and it is close to be optimal when the computational effort of 6000 schedules is computed (Table 11).

Table 9
Performance of the MM-HGA with a computational effort of 1000, 3000 and 5000 schedules generated.

# Schedules	Av.Dev.OS (%)	Max.	St.Dev.	Optimum (%)	CPU-time (s)
J10					
1000	0.26	16.67	1.31	95.34	0.02
3000	0.09	6.67	0.65	97.95	0.05
5000	0.06	6.25	0.55	98.51	0.08
J12					
1000	0.75	17.65	2.24	86.84	0.02
3000	0.31	17.65	1.40	93.97	0.06
5000	0.17	10.00	0.97	96.53	0.10
J14					
1000	1.02	15.79	2.40	80.76	0.03
3000	0.43	8.33	1.38	90.38	0.07
5000	0.32	8.33	1.22	92.92	0.11
J16					
1000	1.30	14.29	2.47	74.18	0.03
3000	0.63	9.68	1.70	86.18	0.08
5000	0.44	9.68	1.39	90.00	0.12
J18					
1000	1.59	18.18	2.93	71.01	0.03
3000	0.79	12.50	1.88	81.88	0.08
5000	0.63	11.11	1.63	84.96	0.13
J20					
1000	2.08	17.65	3.31	64.08	0.04
3000	1.08	14.71	2.26	76.71	0.10
5000	0.87	10.71	1.97	80.32	0.15
J30					
1000	16.65	150.00	26.91	48.74	0.06
3000	15.36	142.31	25.80	51.62	0.13
5000	14.77	138.46	25.08	52.35	0.21

Table 10
Performance of the MM-HGA with the J60 and J120 set of instances and a computational effort of 1000, 3000 and 5000 schedules generated.

# Schedules	J60				J120			
	Av.Dev.LB (%)	Max.	St. Dev.	CPU-time (s)	Av.Dev.LB (%)	Max	St. Dev.	CPU-time (s)
1000	16.57	156.67	26.77	0.12	16.62	154.00	26.54	0.41
3000	14.77	156.67	25.08	0.27	14.48	148.00	24.61	0.91
5000	14.10	143.33	24.45	0.42	13.65	142.00	23.71	1.40

Table 11
New heuristic solution method vs. other heuristics and metaheuristics published in the literature ($J = 10$ and different number of schedules generated).

Heuristic	Av.Dev.OS (%)	Optimum (%)	# Schedules	CPU-time (s)
MM-HGA	0.04	99.07	6000	0.10
MM-HGA	0.06	98.51	5000	0.08
MM-HGA	0.08	98.13	4000	0.07
MM-HGA	0.08	98.13	3000	0.05
Hartmann (2001)	0.10	98.10	6000	–
MM-HGA	0.14	96.83	2000	0.03
Alcaraz et al. (2003)	0.19	96.50	6000	0.19[a]
Bouleimen and Lecoq (2003)	0.21	96.30	–	19.3[b]
MM-HGA	0.26	95.34	1000	0.02
Kolisch and Drexl (1997)	0.50	91.80	6000	–
Özdamar (1999)	0.86	88.10	6000	–

[a] Pentium III with 1.13 GHz and 256 MB RAM.
[b] Pentium with 100 MHz and 32 Mbytes of RAM.

Table 12
Comparison with other heuristics.

Author (year)	Set of instances					
	J10	J12	J14	J16	J18	J20
MM-HGA	0.06	0.17	0.32	0.44	0.63	0.87
Alcaraz et al. (2003)	0.24	0.73	1.00	1.12	1.43	1.91
Józefowska et al. (2001)	1.16	1.73	2.6	4.07	5.52	6.74

% Deviation from optimal solution (5000 schedules).

Besides, a more detailed analysis has been carried out also considering the remaining of the standard sets. In such a case, the comparison is made regarding data reported by Alcaraz et al. (2003) and Tormos and Lova, 2001. In all cases, the computational effort corresponds to 5000 schedules generated. Results appear in Table 12 and evidence the outperformance of the MM-HGA proposed algorithm regardless of the set of instances used.

Finally, the performance analysis of the MM-HGA has been extended to the medium to large Boctor's multimode project instances. These instances are divided into two sets. One set with projects of 50 non-dummy activities and the other one includes project instances of 100 non-dummy activities (Boctor, 1993, 1996a, b). The main characteristic of these sets of instances is the fact that both of them only consider renewable resources. The performance of the MM-HGA is compared with the GA proposed by Alcaraz et al. (2003) clearly improving its results (Table 13) in all cases considered.

5. Conclusions

Applications of project scheduling can be found in diverse economic contexts such as construction engineering, software development, research and development projects, etc. Moreover, project scheduling is important for make-to-order companies where the capacities have been reduced in order to meet lean management concepts. Concretely, due to steadily shortening product life cycles, globalization of markets and decreasing profit margins, industrial projects have to be realized facing tight time and resource constraints that can be modelled as RCPSP. In this highly competitive environment, the development of efficient algorithms able to deal with different execution

Table 13
Comparison with other heuristics for Boctor's instances.

Algorithm	J50		J100		
	Av.Dev.LB (%)	CPU-time (s)	Av.Dev.LB (%)	# Sched	CPU-time (s)
MM-HGA	23.70	0.50	24.85	5000	1.46
MM-HGA	24.89	0.11	26.96	1000	0.34
MM-HGA	25.34	0.07	27.20	500	0.26
Alcaraz et al. (2003)	26.52	0.95[a]	29.16	5000[a]	2.05
Alcaraz et al. (2003)	33.83	0.186[a]	41.85	1000[a]	0.392

[a] Pentium III with 1.13 GHz and 256 MB RAM.

modes for activities shows up as very powerful tools for the decision making process.

Indeed, the MRCPSP is a very challenging problem and in this paper a hybrid GA has been developed that is able to efficiently solve this problem. Firstly, a representation of individuals is proposed that combines both the project characteristics (activities and execution modes) and the key aspects concerning with the generation of the related schedule (schedule generation scheme and direction of application). One of the steps in the solving process of projects involving both renewable and non-renewable resources with limited availabilities is the mode assignment in the initial solution. For this purpose, a new parameter has been designed and its efficiency stated. In the evolution process characteristic of the GAs, fitness function plays a crucial role. A new fitness function has been developed that overcomes the drawbacks of the existing ones. Its effectiveness has been assessed against those of the fitness functions previously developed and the test allows concluding its better behaviour. On the other hand, an improving method able to drastically reduce the project duration has been presented. The effect of the use of the improving method has been analysed and results reported point out its crucial role in the final setting of the algorithm. In fact, the hybrid GA has been tested against the best algorithms published so far and using as benchmark the well known set of instances of PSPLIB as well as the set of projects only including renewable resources reporting in all cases very good results.

Acknowledgements

Authors are grateful to the anonymous referees for their helpful and constructive comments and suggestions.

This work has been partially supported by the research projects TIN2007-29666-E and TIN2007-67943-C02-01 (Min. de Educación y Ciencia, Spain-FEDER), and by the Future and Emerging Technologies Unit of EC (IST priority-6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

References

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.

Blazewicz, J., Lenstra, J., Kan, R., 1983. Scheduling subject to resource constraints: classification and complexity. Discrete Applied Mathematics 5, 11–24.

Boctor, F.F., 1993. Heuristics for scheduling projects with resource restrictions and several resource-duration modes. International Journal of Production Research 31, 2547–2558.

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

Boctor, F.F., 1996b. Resource-constrained project scheduling by simulated annealing. International Journal of Production Research 34, 2335–2351.

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., Möhring, R., Neumann, K., Pesch, E., 1999. Resource-constrained project scheduling: notation, classification, models and methods. European Journal of Operational Research 112, 3–41.

Demeulemeester, E., Herroelen, W., 1992. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. Management Science 38, 1803–1818.

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.

Drexl, A., Grünewald, J., 1993. Nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions 25, 74–81.

Hartmann, S., 2001. Project scheduling with multiple modes: a genetic algorithm. Annals of Operations Research 102, 111–135.

Hartmann, S., Kolisch, R., 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. European Journal of Operational Research 127, 394–407.

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

Józefowska, 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.

Kadrou, Y., Najid, N.M., 2006. A new heuristic to solve RCPSP with multiple execution modes and multi-skilled labor. In: Proceedings of IMACS Multiconference on Computational Engineering in Systems Applications.

Kolisch, R., Drexl, A., 1997. Local search for nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions 29, 987–999.

Kolisch, R., Hartmann, S., 1999. Heuristic algorithms for solving the resource-constrained project scheduling problem. In: Project Scheduling—Recent Models, Algorithms and Applications. Kluwer Academic Publishers, Boston, MA, pp. 147–178.

Kolisch, R., Sprecher, A., 1996. PSPLIB—a project scheduling problem library. European Journal of Operational Research 96, 205–216.

Özdamar, L., 1999. A genetic algorithm approach to a general category project scheduling problem. IEEE Transactions on Systems, Man and Cybernetics—Part C 29, 44–59.

Özdamar, L., Ulusoy, G., 1994. A local constraint based analysis approach to project scheduling under general resource constraints. European Journal of Operational Research 79, 287–298.

Patterson, J.H., Slowinski, R., Talbot, F.B., Weglarz, J., 1989. An algorithm for a general class of precedence and resource constrained scheduling

problems. In: Slowinski, R., Weglarz, J. (Eds.), Advances in Project Scheduling. Elsevier, Amsterdam, pp. 3–28.
Slowinski, R., Soniewicki, B., Weglarz, J., 1994. DSS for multiobjective project scheduling. European Journal of Operational Research 79, 220–229.
Sprecher, A., Drexl, A., 1998. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. European Journal of Operational Research 107, 431–450.
Sprecher, A., Hartmann, S., Drexl, A., 1997. An exact algorithm for project scheduling with multiple modes. OR Spektrum 19, 195–203.
Talbot, B., 1982. Resource-constrained project scheduling with time-resource tradeoffs: the nonpreemptive case. Management Science 28, 1197–1210.
Tormos, P., Lova, A., 2001. A competitive heuristic solution technique for resource-constrained project scheduling. Annals of Operations Research 102, 65–81.
Valls, V., Ballestín, F., Quintanilla, S., 2005. Justification and rcpsp: a technique that pays. European Journal of Operational Research 165, 375–386.