Boletín de Estadística e Investigación Operativa
Vol. 30, No. 2, Junio 2014, pp. 125-149

Investigación Operativa

A hybrid genetic algorithm with transmitted justification for the RCPSP with due dates

Francisco Ballestín

Dpto. Matemáticas para la Economía y la Empresa
Facultad de Economía
Universitat de Valencia

✉ francisco.ballestin@uv.es

Rosa Blanco

Dpto. de Estadística e Investigación Operativa
Facultad de Económicas y Empresariales
Universidad Pública de Navarra

rosa.blanco@unavarra.es

Abstract

This paper deals with two resource-constrained project scheduling problems with due dates in the activities, the TardinessRCPSP and the DeadlineRCPSP. In the TardinessRCPSP the objective is total tardiness minimisation, whereas the DeadlineRCPSP tries to minimise the project length while fulfilling the due date for each activity. In the first part of the paper, an algorithm called hybrid genetic algorithm with transmitted justification (HGATJ) is developed for both problems. This algorithm is based on an algorithm for the classical Resource Constrained Project Scheduling Problem (RCPSP), but uses an extra gene in the codification to control the use of a key technique called justification. Computational results show that the new approach outperforms the heuristics proposed for these problems in a recent paper. The second part of this article adapts the HGATJ to (heuristically) solve the multi-objective problem combination of the TardinessRCPSP and the DeadlineRCPSP. A path-relinking phase is introduced in the algorithm to produce a better Pareto front.

Keywords: Resource-Constrained project scheduling, Due dates, Heuristic algorithms, Multi-Objective.

AMS Subject Classifications: 90B10, 90C27, 90C29

© 2014 SEIO

1. Introduction

When executing a project, resources have to be allocated to precedence-related activities so that one or several objectives are met. More often than not, some tasks should or must be finished by a certain time, a due date or deadline. This paper proposes an algorithm for two project scheduling problems involving due dates, the TardinessRCPSP and the DeadlineRCPSP. In both problems, precedence relations are of the type finish to start and activities consume several types of renewable resources, i.e., resources that are available at a constant and fixed rate throughout the project duration. In the TardinessRCPSP, the objective is total tardiness minimisation, whereas in the DeadlineRCPSP the due dates are strict (deadlines) and the objective is makespan minimisation. The second problem is much harder than the first, in the sense that finding a feasible solution is already NP-hard (Garey and Johnson, 1979). The DeadlineRCPSP is a generalisation of the well-known Resource Constrained Project Scheduling Problem (RCPSP), with the fulfilment of the due dates as extra restrictions. There are many other extensions of the RCPSP, we refer to the surveys of Özdamar and Ulusoy (1995), Herroelen et al. (1998) and the books by Neumann et al. (2003) and Józefowska and Weglarz (Eds.) (2006). Although being an essential characteristic of real projects, little attention in relation with the RCPSP has been paid to due dates. Few researches have studied due dates in related problems like Vanhoucke et al. (2001) in the RCSPWET (resource-constrained project scheduling problem with weighted earliness-tardiness) or Viana and Pinho de Sousa (2000) in a problem with multiple modes per activity, non-renewable resources and three objective functions. In Kis (2005), the resource usage of each activity may vary over time proportionally to its varying intensity, whereas in Drezet and Billaut (2008), activity requirements are time-dependent and employees (resources) have different skills. Klein and Scholl (2000) develop an exact procedure for the generalized RCPSP by considering non-negative minimum time lags and time-varying resource availabilities apart from release and due dates.

The large number of referenced papers in the review of Gordon et al. (2002) show that machine scheduling problems including due dates are of permanent interest. All these factors support the further study of due dates in the RCPSP.

TardinessRCPSP and DeadlineRCPSP are denoted by PS | prec | T and PS | temp | Cmax respectively in the notation of Brucker et al (1999) or m,1 | cpm | T and m,1 | cpm,δ_j | Cmax in the notation of Herroelen et al (1998), with T given by

$$\mathrm{T} = \sum_{\mathrm{j}=2}^{\mathrm{n}-1} \max\left(0, \mathrm{s}_j + \mathrm{d}_j - \mathrm{dd}_j\right)$$

where n-2 is the number of real activities and 1 and n are the dummy source

and sink of the project. Besides, s_j, d_j and dd_j are the starting time, duration, and due date of activity j, respectively. S is the solution or schedule formed by the starting times s_i. We suppose that there are no due dates assigned to the dummy activities 1 and n.

In Ballestín et al. (2006), the TardinessRCPSP and DeadlineRCPSP were studied. The performance of well-known RCPSP heuristics – priority rules, sampling procedures and metaheuristics – was compared with the results of new versions that took due dates into consideration. One of the techniques that exhibited good results was the justification. The justification of an activity to the right (left) consists in scheduling the activity as late (early) as feasible, taking into consideration precedence and resource restrictions. The justification to the right (left) of a schedule S calculates a new schedule S' by justifying the activities to the right (left) in a certain order. A schedule S can be first justified to the right, obtaining S' and then S' can be justified to the left, obtaining S". S" is then called the Double Justification of S, DJ(S). The justification, also called forward-backward pass or improvement, was introduced in 1964 by Wiest and has been used, among others, by Li and Willis (1992) and Tormos and Lova (2001). Valls et al. (2005) showed the potential of DJ by incorporating it into a wide range of algorithms for the RCPSP, increasing their solution quality while maintaining the number of schedules calculated. Nowadays many of the best heuristic algorithms for the RCPSP apply some kind of justification. One of these top algorithms is HGA (Hybrid Genetic Algorithm), presented in Valls et al. (2005).

There are several types of justification, depending on the order chosen for the activities to be justified (Valls et al., 2006). The justification used in the RCPSP is the *justification by extremes*. At each of the n iterations of the procedure to the right (left), the activity with the largest finish time (smallest start time) is selected to be justified. In the *justification by eligibles*, Valls et al. (2006), other priority rules to select the activity to be shifted are allowed. Once due dates are defined, new rules can be defined to take advantage of the new information. Ballestín et al. (2006) compared several tailored rules and showed that the use of the best rules outperformed using the justification by extremes.

The contribution of this paper is threefold. First of all, to create a new form of justification, called justification with due dates, that goes further in the use of the information given by the due dates than the justification by eligibles (section 2). Secondly, to incorporate this technique in an adaptation of HGA (section 3), creating HGA with transmitted justification (HGATJ). Section 4 contains the results of this algorithm and the comparison with the best algorithms in the literature. Finally, in many real applications, the manager still wants to minimise the makespan even if not all due dates can be fulfilled. I.e., we are dealing with a multiobjective problem where we want to minimise the total tardiness and the makespan. The third goal of the paper is to develop a heuristic algorithm

based on HGATJ capable of providing (an approximation of) the Pareto front for this problem. Several enhancements have been introduced in a standard adaptation. Section 5 describes the standard and improved algorithm, whereas section 6 compares their approximations of the Pareto front.

2. The justification with due dates

The justification has been adapted to other generalisations of the RCPSP, e.g. the RCPSP with preemption (Ballestín et al., 2009) and the multi-mode RCPSP with maximal time lags, the MRCPSP/max (Barrios et al., 2009). In each case, the additional characteristics of the problems could be used to modify the justification and produce a better improvement method. As commented upon above, the justification by eligibles with specific rules proved in Ballestín et al. (2006) its superiority in both TardinessRCPSP and DeadlineRCPSP to the usual justification by extremes. However, due dates information can be further exploited to achieve an even bigger improvement. In this section we create a new way of justification, the justification with due dates.

Let S be a schedule of the problem and i an activity starting in s_i. We say that i is delayed in S if $s_i + d_i > dd_i$. Non-delayed activities are then those activities that finished no later than their due dates in S ($s_i + d_i \leq dd_i$). We will omit the "in S" if the solution we are referring to is clear. Clearly, if an activity i is non-delayed in S, it does not affect (negatively) the objective function of S in the TardinessRCPSP, since $(s_i + d_i - dd_i)^+ = 0$. In the process of justifying S to the right, obtaining S', there is an iteration in which we justify i to the right. In that moment, in all types of justification considered up to now, we schedule i as late as possible, let us say to s_i'. If $s_i' + d_i > dd_i$, and S and S' share the same makespan, i will negatively affect T(S'). This setback may be overcome when we justify S' to the left and obtain S", because i may start in S" before it does in S' or even in S. However, this is not always the case, and we can end up with T(S") >T(S). This was already noted in Ballestín et al. (2006), where an example was shown. Analogously, the usual justification may transform feasible solutions in the DeadlineRCPSP into unfeasible ones.

But the goal/restriction of the problem, $s_i + d_i \leq dd_i$, suggests a way to avoid this drawback. When we justify a non-delayed activity to the right, we will force the activity i to be justified only until its due date, so that $s_i' + d_i = dd_i$ at the most. We will call these restrictions *non-delayed restrictions*. They do not apply to delayed activities because they already finish after their due dates. But moving a delayed activity to the right affects negatively the objective function or the feasibility in our problems, too. We can also impose restrictions on the delayed activities for the justification to the right, called *delayed restrictions*: when justifying to the right, do not shift delayed activities at all. Summing up, we can introduce n restrictions in the justification, one for each activity, no

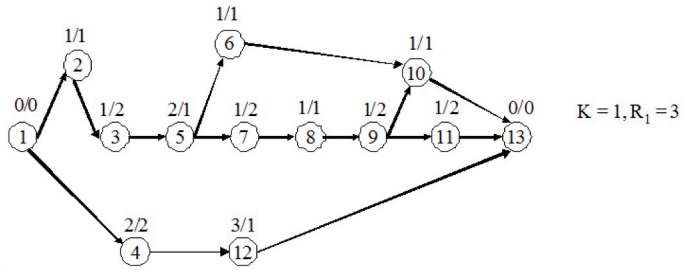

Figure 1: Project for the justification with due dates

matter which schedule we are working with. We define the *justification with due dates* as the justification that works with these n restrictions. As far as we know, this is the first time that restrictions are imposed in the justification.

The following result holds: if the justification with due dates is applied, then $T(DJ(S)) \leq T(S') \leq T(S)\ \forall S$. So, this procedure is an improvement function in the TardinessRCPSP and an improvement function that transforms feasible solutions into feasible solutions in the DeadlineRCPSP. As commented upon above, this contrasts with the non-adapted justification, which may worsen solutions in the former problem or make them unfeasible in the latter.

The new justification is able to improve schedules that cannot be improved by the other commented justification types. The schedule S of Figure 2 is active in the project depicted in Figure 1. We consider the problem with $dd_6 = 7$ and $dd_{11} = 8$ as the only non-trivial due dates. With these due dates we have $T(S) = 1$. If we use the justification with due dates and justify S according to the activities' ends, we justify 6 until its due date, finishing it in 7. Then we can justify 12 to [5,8], 4 to [3,5] and 2 to [1,2], obtaining S' (Figure 3). Since the first activity of S' begins in 1, we can subtract 1 from the beginning from every activity. We obtain a schedule where the end of 11 is 8 (and activity 6 remains non-delayed), so its objective function in the TardinessRCPSP is 0. In the DeadlineRCPSP, S is unfeasible whereas S' is feasible. The only schedule S" attainable by the non-adapted justification to the right (Ballestín, 2002) is depicted in Figure 4. No solution better than S can be reached from S' when justifying to the left.

Nevertheless, the justification with due dates also presents some drawbacks, as shown in the following example. We consider the project in Figure 5, with $dd_2 = 2$, $dd_4 = 1$ and $dd_5 = 2$ as the only non-trivial due dates. The sequence S_2 of figure 6 is active for this project. If we apply the justification by extremes to S we obtain S_2', the schedule of Figure 7. The objective function of S2' in the TardinessRCPSP is $0 < 1 = T(S2)$, once we have subtracted one from the beginning of every activity. In the DeadlineRCPSP, S_2' is feasible while S_2 is not.

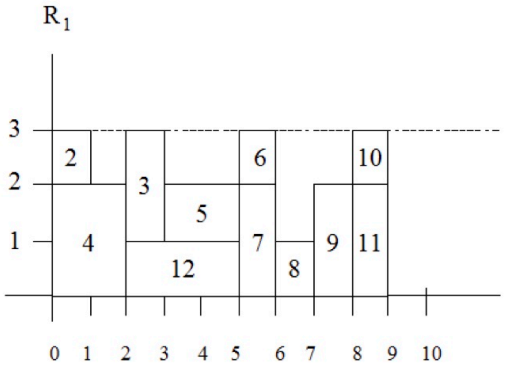

Figure 2: Active schedule S for the project of Fig. 1

Nonetheless, the justification with due dates is not able to move any activities in S_2, due to resource, non-delayed or delayed restrictions. Hence, the justification with due dates is not able to improve S_2.

This exemplifies the major drawback of the new procedure. The restrictions on the activities restrain, sometimes too much, the movements of activities to the right, and consequently the new schedule and DJ(S) may not be very different from the original. In these cases an improvement in the objective function is very unlikely. To take advantage of the due dates but without being too restricted we introduce the concept of *justification with due dates with k restrictions*. It consists in applying the delayed and/or the non-delayed restrictions only to k out of the n activities, whereas the rest of the activities are fully justified. We will suppose that the k activities to which we impose the restrictions are randomly chosen. Regretfully, the best k depends on the instance we are working with, usually being a small (large) k better for instances with tight (loose) due dates. Therefore we are going to use a self-adapting justification, which will be introduced in section 3.

A new justification for the DeadlineRCPSP

The problem of finding feasible solutions is simple in the TardinessRCPSP but NP-hard in the DeadlineRCPSP. Therefore we have introduced a new feature in the justification with due dates to facilitate the finding of feasible solutions. Let S be an unfeasible schedule. Until now, the justification never justifies the last activity, n. This is equivalent to fix n in makespan(S) and to justify the rest of the activities in the interval [0, makespan(S)]. This method is inherited from the RCPSP, but the makespan is not so important while S is unfeasible. The novelty we propose is to fix $dd(n) := \max\{\text{makespan}(S), \max\{dd_i , i < n\}\}$ and to justify the activities with the justification with due dates (with a given k) in

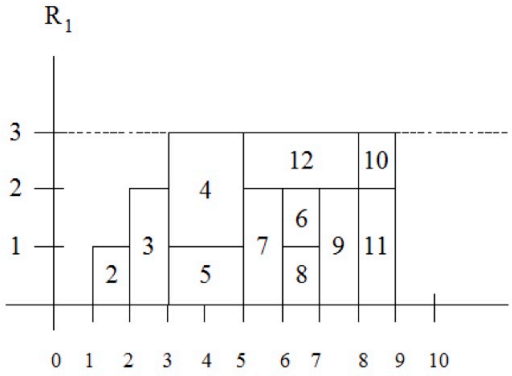

Figure 3: Justification with due dates of S to the right

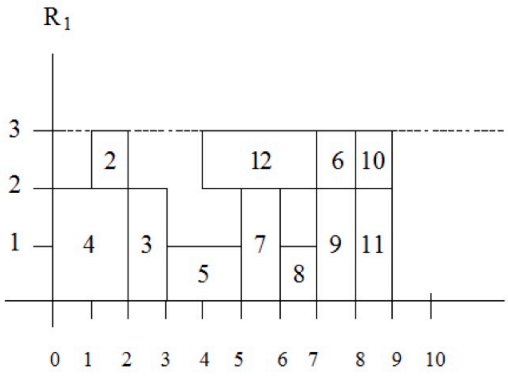

Figure 4: Usual justification of S to the right

[0, dd(n)]. The dummy activity n does not have a due date dd_n and therefore we work with dd(n). We will call this justification *maxdd-justification*, because it justifies activities until the maximum of due dates (if it is greater than the makespan). When dd(n) is equal to makespan(S), the new strategy is the same as the justification with due dates. In the rest of the cases we have a bigger interval to shift activities, thus creating gaps to shift activities that did not exist previously.

We consider the project in Figure 8, with $dd_2 = 3$ and $dd_3 = 1$. The schedule S_3 of Figure 9 is unfeasible for the DeadlineRCPSP. S_3 cannot be changed by any of the justification types we have seen until now. So, $DJ(S_3) = S_3$ for every justification. However, if we justify to the right according to the new strategy, we fix $n = 4$ in $t = 3 = \max\{dd_i, i < n\} = dd_2$. Thus, we can move

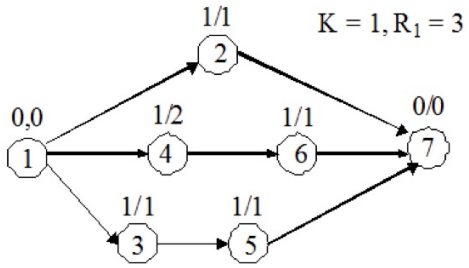

Figure 5: Project for showing drawbacks of justification with due dates

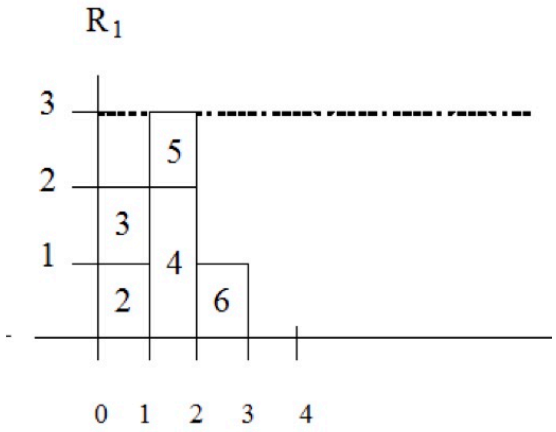

Figure 6: Active Schedule S_2 for project in Fig. 5

2 to [2,3], obtaining the schedule of Figure 10. Subtracting 1 unit from the beginning of both activities, we obtain a feasible (and optimal) schedule for the DeadlineRCPSP.

This new technique also has its drawbacks. The justification to the right often shortens the makespan of a schedule and this may lead to a decrease in the delay of the delayed activities. If we use the justification with k due dates, justifying in [0,dd(n)] instead of [0,makespan(S)] decreases the possibility of shortening the makespan of a given schedule S, even taking into account the posterior justification to the left. So, this new technique is not very effective if we want to minimise the makespan and we have already found a feasible solution. Therefore we have incorporated the new technique in the algorithm for the DeadlineRCPSP, but only until it finds a feasible solution. HGATJ

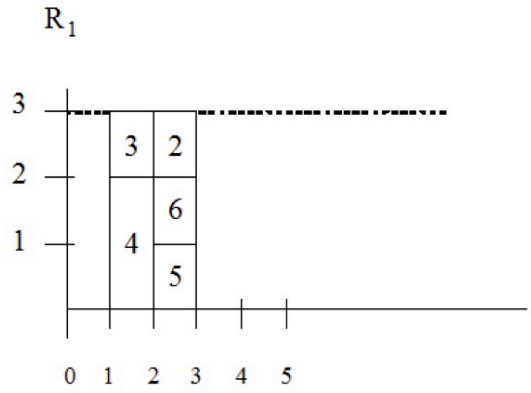

Figure 7: Justification by extremes of S_2

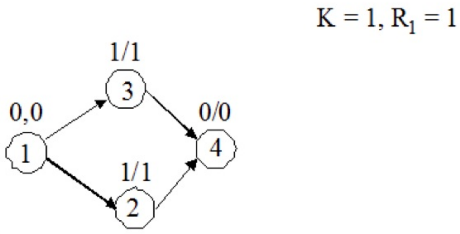

Figure 8: Project for maxdd-justification

for the DeadlineRCPSP works as follows. It applies the justification with due dates (with the k transmitted via the solutions) fixing n in dd(n) until it finds a feasible solution. Afterwards, it uses the usual justification with due dates (with the correspondent k), fixing n in makespan(S).

3. HGA with transmitted justification

3.1. Description of HGA

Since HGA is already described in Valls et al. (2007), we are only going to summarise it here. First we present the global scheme in Figure 11.

In each generation of the first phase, the size of the population is constant and equal to *POPsize* where *POPsize* is an even integer. In each generation of the second phase, the size of the population is constant and equal to *POPsize/2* if this number is even, equal to *POPsize/2 –1* otherwise. The number of schedules generated in each phase is upwardly limited by *nsche/2*. Given nsche and

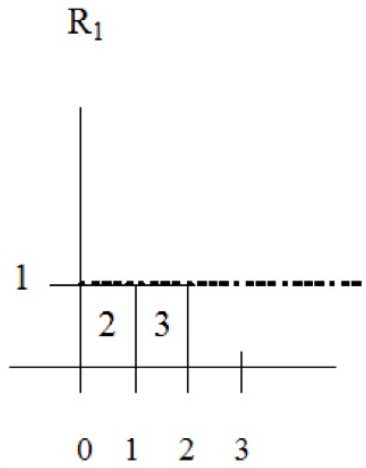

Figure 9: Unfeasible Schedule S_3 for project of Fig. 8

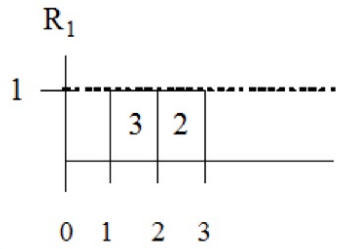

Figure 10: Maxdd-justification applied to S_3

POPsize, it is easy to calculate niter so that this limit is not exceeded.

We are going to describe the essential aspects of HGA in the following subsections.

Codification and S-SGS

To simplify the exposition of the algorithm, we will assume that HGA works with so-called activity lists, although it works with a special type of them. An activity list is a permutation of the activities that fulfils the precedence relationships. The *serial schedule generation scheme* (S-SGS, Kolisch, 1995) transforms an activity list λ in a schedule $S(\lambda)$ by scheduling the activities in the order of the list λ at their earliest precedence and resource-feasible start time. This procedure generates active schedules. As happens in the RCPSP, there is always an optimal active schedule for the TardinessRCPSP and the DeadlineRCPSP (if

1. Create an initial population.
2. For [i =1,niter]
  2.1. Select pairs of individuals (parents).
  2.2. Produce children by applying:
    2.2.1. The peak operator to each pair of parents.
    2.2.2. The mutation and the double justification operators to each child.
  2.2. Add the children to the population.
  2.2. Reduce the population by selection.
3. Create a neighbour's population from the best solution so far.
4. Apply step 2 to the new population.
5. Return the best solution found.

Figure 11: The HGA scheme

there are feasible solutions). Consequently, it is enough to work with activity lists and the S-SGS to search in the correct solution space.

Peak crossover

The objective of the peak crossover operator is to exploit the knowledge of the RCPSP to identify and combine those good parts of the solutions that have really contributed to its quality. There is a clear relationship between a higher/lower utilisation of resources throughout the project and the makespan of a solution. A low utilisation of resources usually means scheduling the activities almost in a serial way, something which lengthens the end of the project. The high use of resources is therefore a desirable characteristic in a schedule. Each time interval with a high use of resources (peak) corresponds to a set of activities that are scheduled together, or equivalently, to a part of the activity list, a subactivity list. The peak crossover selects a mother and a father. Then it detects the peaks in the mother and introduces them in the empty activity list of the daughter. The father determines the position of the remaining activities that are put before and after each peak. The son is created exchanging the roles of the mother and the father.

Initial population

We employ the regret based biased random sampling method, Drexl (1991), together with the LFT (Latest Finish Time) priority rule and $\alpha = 1$ to obtain POPsize schedules. The DJ is applied to each of the schedules.

The 2-phase strategy

Given an activity list λ, the procedure we have implemented to construct a population of nearby activity lists is a simplified version of that used in Valls et al. (2003). The procedure generates $POPsize/2$ (or $POPsize/2$ - 1) schedules with $\beta = 1$ - 20/n by applying the *β-biased random sampling method*, *β-BRSM*, to λ. The neighbour's population is obtained by first applying the double justification

operator to each of the generated schedules and then obtaining their activity list representations.

β-BRSM can be described as follows. Given an activity list λ, the method makes use of the S-SGS. The following strategy is employed to select, in each iteration, the next activity j to be scheduled. A random number p∈(0,1) is generated. If p < β, then j is the eligible activity with the lowest position. An activity is eligible if all its predecessors have been chosen. Otherwise, j is one of the other eligible activities selected by biased random sampling (Kolisch and Hartmann (1999)), and employing the positions as priority values in order to obtain the selection probabilities. Note that the parameter β controls the extent to which the new vector differs from the original.

3.2. Changes in the original algorithm

We have introduced two changes in the HGA to cope with the problems with due dates, two elements that proved their usefulness in Ballestín et al. (2006): the priority rule used in the calculation of the initial population and the justification.

Calculation of the initial population

As we have commented upon above, LFT is the priority rule employed in the sampling method in the first step of HGA to calculate the initial population. This rule may be useful for the RCPSP, but does not take into account due dates. The priority Earliest Due Date EDD schedules first those activities with smallest due date. We have applied it with the Parallel SGS, another SGS different from the Serial SGS. At each iteration of the Parallel SGS (Kolisch, 1995), a decision time t is considered. The procedure starts as many activities as possible at t, following some priority rule – in our case the EDD rule. The first decision time is 0 and the rest of the decision times are the finish times of the activities. In step 1 of HGATJ, half of the solutions are created with LFT + Serial SGS, the rest of them with EDD + Parallel SGS.

Justification

HGATJ applies the justification with due date with k restrictions instead of the justification by extremes of HGA. During the first computational tests, we noted that for some instances it is better to use the justification with due dates with big k's – instances with loose due dates – and for others it is better low k's – instances with tight due dates. Summing up, the k is instance dependent. We want an algorithm capable of adjusting the k to the instance it is working with, without any previous information of the instance. To solve this difficulty the codification of each solution in the HGATJ carries an extra gene, called the justification gene. This gene contains, in general, the k with which the solution S has been obtained after a double justification. Every time a solution S is created, it is assigned a certain k; we'll explain later on how. When the justification is

applied to S, the justification with due dates with k restrictions is applied, thus obtaining S' = DJ(S). If T(S') < T(S), the justification gene of S' is set up to k. So, as we have said, in general the justification gene corresponds to the k with which the solution has been obtained. If T(S') $\geq$ T(S), the justification gene of S is randomly calculated. With this action we dismiss the original k. The justification gene given to a solution S not obtained by justification depends on how it is built. If S is the son (daughter) of a crossover between a father and a mother, S inherits the justification gene from the father (mother). In the initial population of the first and second phase, it is randomly calculated. We say that the number of restrictions in the justification is transmitted through the solutions and call the algorithm HGATJ, HGA with transmitted justification.

If a very good solution is obtained with a certain k, it is usually because the justification with due dates with this k works well in the instance. With our method, this k is transmitted to the children of the solution, as long as it survives in the population. Since it is a very good solution, it will survive many iterations. If a k does not produce good results, it will soon disappear from the population, because we store another gene instead. In order to avoid the prevalence of one or a few k's, we have also implemented a mutation. It consists in changing the extra gene with a certain probability before applying the S-SGS. In the computational results we use 0.05.

Another way to implement the justification

In HGATJ, "only" the level of tightness in the justification is transmitted from parents to children. We could go further and also transmit the specific activities which can be shifted freely and which ones should not go further than their due dates. To study this option, we have run a version of HGATJ with an extra binary vector η for each codification. This vector has n components; η(i, S) is the i-th component of the vector associated to solution S. If η(i, S) = 1, when we justify S to the right, we will impose a delayed/non-delayed restriction on i if i is delayed/non-delayed. If η(i, S) = 0, activity i will be shifted freely.

We have tried two versions of the algorithm. In the first one, the daughter inherits η from the mother and the son inherits η from the father. In the second version, we apply a two-point crossover to the vectors η of the mother and the father, obtaining the vectors for the daughter and son. In both cases, we randomly calculate these justification vectors at the start of the algorithm. Besides, we apply a mutation that changes some of the genes of η with a certain probability. Preliminary results showed that no better results were obtained with these versions with regards to HGATJ, and therefore we have not incorporated these versions in the next section.

4. Computational results

In Ballestín et al. (2006) a procedure was developed to introduce due dates in RCPSP instances. The goal was to produce instances with different levels of difficulty in the fulfilling of the due dates. Concretely, three sets based on the standard set j120 were created: the loose, medium and tight set. The set j120 consists of 600 projects with 120 activities, generated under a full factorial experimental design with several problem parameters (Kolisch and Sprecher, 1997). In the loose set, it is easy to fulfil most of the due dates at the same time and only a limited subset of activities are going to be tardy in any schedule. In the tight set, on the contrary, many activities are going to be tardy in any schedule. The medium set contains instances in between. We have worked with a total of 1670 instances.

All algorithms of this section will be run until 5000 schedules are generated. This is the most usual upper limit imposed when comparing state-of-the-art heuristic algorithms for the RCPSP. To compare algorithms in the TardinessRCPSP, we use the average of the total tardiness over each of the three instance sets ("loose set", "medium" and "tight set"). For the comparison in the DeadlineRCPSP, the number of instances where a feasible solution has been found is included ("# feas. sol. in DeadlineRCPSP").

4.1. Comparisons of the different justification types

We have run our algorithm with the different versions of the justification introduced in sections 2 and 3. Apart from our standard algorithm HGATJ, the following versions are considered: justification with due dates only with the delayed restrictions (3^{rd} row), justification with due dates only with the non-delayed restrictions (4^{th} row), an extreme version with 0 restrictions (5^{th} row) and an extreme version with n restrictions (6^{th} row). Note that "0 restrictions" correspond to the justification by eligibles, because it does not used the justification by due dates. Besides, the version "n restrictions" correspond to applying the justification with due dates. Finally, in the last row, the HGA with the max-ddn justification applied to every schedule is run.

	loose set	medium set	tight set	# feas. sol. in DeadlineRCPSP
HGATJ	6.43	177.58	1100.06	424
Only delayed restrictions	8.41	189.76	1152.16	382
Only non-delayed restrictions	8.13	197.26	1159.62	398
0 restrictions	11.30	195.03	1088.31	348
n restrictions	6.76	194.62	1172.68	426
maxdd-justification	6.42	177.19	1100.04	425

Table 1: Results of different versions of HGATJ

As we can see in Table 1, restraining the justification gives very good results in the loose set. In the medium set, restraining the justification too little or too much worsens the solution quality. The tight set is where less difference occurs among algorithms. The reason for this behaviour is that the justification with due dates cannot change the solutions much in the tight set, since many activities are delayed or very near their due dates. Nevertheless, using the justification with 0 restrictions does not improve the results much. All in all, the best algorithms for the TardinessRCPSP are HGATJ and HGATJ with maxdd-justification. We will see in section 4.3 why we have chosen HGATJ as our final algorithm.

4.2. Comparisons with state-of-the-art heuristic: TardinessRCPSP

In this section we are going to compare HGATJ with the best algorithm in Ballestín et al. (2006). Table2 contains the results in the TardinessRCPSP, along with the improvement HGATJ attains (= (Ballestín et al. – HGATJ)/HGATJ).

	total tardiness average			% improvement		
	loose set	medium set	tight set	loose set	medium set	tight set
Ballestín et al. (2006)	11.41	199.02	1119.18	77.33%	12.07%	1.74%
HGATJ	6.43	177.58	1100.06	-	-	-

Table 2: Improvement from HGATJ in the TardinessRCPSP

The algorithm HGATJ clearly outperforms the other algorithm in all the sets. We can observe that the improvement is very important in the loose set and reasonably important in the medium set. The smallest improvement happens in the tight set, where we have seen that the justification with due dates works worse.

4.3. Comparisons with state-of-the-art heuristic: DeadlineRCPSP

Algorithm	# feas. solut. in DeadlineRCPSP	% impr. in #feas. sol	makespan deterioration
HGATJ	424	-	-
H+DJ R1-L6	282	50.35%	0.37%
H+DJ R4-L6	336	26.19%	1.93%
H(2)+DJ R1-L6	342	23.98%	1.80%
H(2)+DJ R4-L6	348	21.84%	2.39%

Table 3: Comparisons in DeadlineRCPSP

Table 3 compares HGATJ with the best algorithms in the DeadlineRCPSP. It is a little more difficult to compare algorithms in this problem, because there are two measures to take into account, the number of feasible solutions and the makespan achieved in those instances. This is the reason why we have taken four algorithms from Ballestín et al. (2006); none outperforms the others in both measures. For example, H+DJ R1-L6 is the best algorithm in makespan and worst in number of feasible solutions.

The third column of the table contains the improvement percentage in the number of feasible solutions obtained by HGATJ. In each cell of the fourth column we only take into account instances where both HGATJ and the corresponding algorithm have obtained feasible solutions. For each of these instances, we calculate the relative deterioration in the makespan, i.e.,

$$\frac{\text{makespan(algorithm)} - \text{makespan(HGATJ)}}{\text{makespan(HGATJ)}}.$$

The number in the cells is the average of these quotients.

The first conclusion of the table is the increase in the number of instances where a feasible solution is reached. For example, HGATJ doubles H+DJ R1-L6 in number of instance with feasible solutions. Concerning the second measure, an improvement around 2% is obtained with the rest of the cases. Except in the case of H+DJ R1-L6, which obtains in this measure similar results than HGATJ. Summing up, HGATJ clearly outperforms the four algorithms.

If we compare the version maxddn-justification with the four algorithms, the makespan deterioration is -0.59%, 0.68%, 0.57% and 1.12%, respectively. Hence, in the second measure it is better not to use maxddn throughout the whole algorithm. This is the reason why we have not used this justification in HGATJ.

5. An algorithm for the multiobjective problem

5.1. Introduction

Having studied the TardinessRCPSP and the DeadlineRCPSP, it is natural to go further and try to offer solutions for a combination of both problems, solutions with a low tardiness and makespan. This cannot be done by aggregating the two functions in one and solving the corresponding problem, since the total tardiness can go from 0 to one or two orders of magnitude of difference with the makespan. We will call the *multiobjective tardiness-makespan RCPSP* the problem with the restrictions of the RCPSP and two objective functions: total tardiness and makespan. Note that we do not impose the due dates as restrictions because (1) if we must satisfy them, we obtain the DeadlineRCPSP, (2) in many practical cases not all due dates are going to be met but the manager still wants to meet as many as possible while having a small makespan and (3) in some cases managers

would sacrifice the fulfilment of one or a few due dates if s/he can have a (much) better makespan.

To solve the multiobjective tardiness-makespan RCPSP means to calculate the non-dominated solutions. A feasible solution S is *efficient* if there is no other feasible solution strictly better than S for at least one criterion and not worse in the remaining criteria. If S is efficient, then y = (f_1(S), f_2(S), ..., f_p(S)) is called *non-dominated*, if f_i is the i-th objective function. A *minimal complete set* contains one decision vector for every non-dominated objective vector. In general, a multiobjective combinatorial optimisation problem is considered as solved if a minimal complete set – sometimes called Pareto front – is calculated. We have adapted HGATJ into MOHGATJ to calculate an approximation of the Pareto front.

Despite the fact that the RCPSP is an inherently multi-objective problem (Viana and de Sousa, 2000), few papers have been published in this field. In a recent paper, Ballestín and Blanco (2009) revised most of those papers and developed different metaheuristic methods for the regular case and for one case with regular and one non-regular objective function. The regular case occurs when all objective functions are regular. A regular objective function is a non-decreasing function of the activity start times (in the case of a minimisation problem). The makespan and total tardiness are examples of regular objective functions. Resource levelling, weighted earliness-tardiness and stability are examples of non-regular objective functions.

The best algorithm in Ballestín and Blanco (2009) was a combination of DJGA (Valls et al., 2005) and NSGA2 (Deb et al., 2000). DJGA+NSGA2 uses activity lists as codifications, the S-SGS as decoder, the two-point crossover and the justification by extremes as improvement method. DJGA+NSGA2 was also applied to the multiobjective tardiness-makespan RCPSP and therefore it will be the keystone for MOHGATJ.

We have created a multiobjective algorithm, MOHGATJ, by incorporating the management of the population of NSGA2 into HGATJ. That is, HGATJ is still responsible for the calculation of every solution, but NSGA2 decides about the survival of individuals. NSGA2 classifies the population into layers, being the first layer the non-dominated solutions. At each iteration of MOHGATJ, a new population is created with the current one and every solution created (before and after the justification) at that step with HGATJ. Then, the layers are formed and the worst layers are erased until a certain size is reached. To distinguish inside a layer, the so-called crowding distance is applied.

Another change occurs in step 3 of Figure 11, when the population for the second phase is created. There, in the HGATJ, the best solution found so far is used to calculate that population. In the MOHGATJ, however, every non-dominated solution found so far contributes to create the new population. To build a solution, one of these non-dominated solutions is randomly chosen and

the β-BRSM (section 3.1) is applied to the activity list of that solution. The outcome S and DJ(S) are inserted in a set. The population for the second phase is obtained by trimming this set with NSGA2.

As stated in Ballestín and Blanco (2009), a heuristic algorithm that works with activity lists and the S-SGS can find a complete set of efficient solutions in any MORCPSP with regular objective functions. So, MOHGATJ is searching in the correct solution space when solving the multiobjective tardiness-makespan RCPSP.

5.2. Enhancements in the standard algorithm: the third phase

We have introduced two enhancements in MOHGATJ. The first one has to do with diversity. We do not allow an individual to enter in the population if all its objective values coincide with those of an element of the population. Besides, in the construction of the population for the second phase of MOHGATJ, we only calculate solutions from individuals with different objective functions. In the computational results we will see that comparing solutions (genotype) instead of objective functions (phenotype) does not improve the algorithm.

The second enhancement consists in adding a third phase to MOHGATJ. In this phase, we exploit the efficient solutions found in the algorithm. After the evolutionary algorithm, the approximation of the Pareto set contains several, usually few, schedules with different makespan and tardiness values. Presumably, all of them contain good features, because these features have lead to non-dominated solutions. This seems a perfect scenario to apply Path-Relinking (PR, Glover and Laguna, 1999) to these solutions, trying to find new ones that combine those characteristics. PR is used to connect elite solutions obtained by other (meta)heuristic techniques. PR starts from an initial solution and gradually incorporates attributes from a guide solution. Thus, each new solution resembles more the guide solution and less the initial one. The method to combine solutions is usually different from other methods applied during previous phases. In our case, we wanted an operator capable of obtaining solutions near a model solution, but with some characteristics of a guiding solution. To accomplish that goal we use the percentage operator, Valls et al. (2005), a generalisation of β-BRSM to deal with two solutions. Given a percentage *percent*, and two activity lists λ and χ, this operator obtains an activity list μ in the following manner. At each of n stages, the set of eligible activities is calculated and a random number (p) is generated between 0 and 1. If p $<$ percent, then an eligible activity is selected with the lowest order in λ. In this case it behaves like the β-BRSM. If p $\geq$ percent, however, the percentage operator chooses the eligible activity with the lowest order in χ. Subsequently, the greater percent, the more weight χ will have in the creation of μ and the less weight λ will have.

We have tried different versions of the third phase. All of them work with an elite set ES. At the beginning of the procedure, ES is formed with the efficient

solutions of previous phases. Following the idea of diversity commented upon above, we do not allow solutions in ES with the same objective functions. However, we do introduce dominated solutions until | ES | = 5. The inclusion rule is the one followed by NSGA2.

A) PR I

We repeat the following loop until the limit in the number of schedules is reached.

1. Choose λ and χ randomly from ES.
2. Initial = λ.
3. Do
   a. μ = %Operator(Initial, χ, 5%).
   b. Calculate S = S-SGS(μ), S' = DJ(S).
   c. ES = ES $\cup$ {S, S'}.
   d. Initial := μ.
4. Until $\mu = \chi$.
5. Eliminate non-efficient solutions from ES.

PRI randomly chooses two solutions from ES, λ and χ. The percentage operator is applied to them with percent = 5%, obtaining an activity list μ. Then S = S-SGS(μ) and DJ(S) are calculated. Theoretically μ is more similar to χ than λ. Afterwards, we change our initial solution and apply the percentage operator to μ and χ, again with percent = 5%. We repeat these steps until S(μ) = S(λ). All solutions created are introduced in the eligible set and we eliminate the dominated solutions.

B) PR II

We repeat the following loop until the limit in the number of schedules is reached.

1. Choose λ and χ randomly from ES.
2. μ = %Operator(Initial, χ, 5%).
3. Calculate S = S-SGS(μ), S' = DJ(S).
4. ES = ES $\cup$ {S, S'}.
5. Eliminate non-efficient solutions from ES.

Some of the μ's obtained in step 3.a in PRI may be worse than λ and χ. So, they might not be good solutions to guide the PR. To investigate this issue, we have introduced the following change in the algorithm. Once we have applied the percentage operator for the first time to λ and χ, we introduce S and DJ(S) in ES if they are non-dominated. Then, we select a new initial and guide solutions and continue the algorithm.

C) β-BRSM

We have also tried a third phase that does not combine solutions, but applies β-BRSM to solutions of ES. We work with β = 1-5/n, with which we intensify instead of diversify.

5.3. Comparison of the Pareto front of the algorithms

The comparison among multiobjective algorithms is still an issue. In several papers, e.g. Deb et al. (2000), a maximum in the number of function evaluations is imposed. This is approximately also done to compare heuristic algorithms in the uniobjective case in the RCPSP (see Hartmann and Kolisch, 2000), where the number of created schedules is bound. Here we will also stop every algorithm after a certain number of schedules created. We will compare algorithms with the distance of their final sets from a reference set (Czyzak and Jaszkiewicz, 1998). The Pareto approximation set of the union of sets obtained by the different algorithms is used as the reference set. To study the significant differences among algorithms we will use the test of signs (Fisher, 1925, Dixon and Mood, 1946).

Table 3 contains the comparison of the different versions with a limit of 5000 schedules. HGATJ+NSGA2 is just the algorithm HGATJ which handles the population with NSGA2. The versions "without repetitions" include the diversity condition in HGATJ+NSGA2. The three versions MOHGATJ include the diversity condition and one version of the third phase.

	Distance to ref. set		Distance to ref. set
HGATJ+NSGA2	0.1679	MOGHATJ (β-BRSM)	0.1587
without repetitions – genotype	0.1622	MOGHATJ (PR I)	0.1527
without repetitions – phenotype	0.1611	MOGHATJ (PR II)	0.1537

Table 4: Comparisons of different versions of MOHGATJ

Not allowing individuals with the same objective functions (version "without repetitions") slightly, but significantly, improves the results. Comparing solutions with their start times instead of with the objective functions does not add much, however, to the quality of the algorithm. We have therefore incorporated the first option into the rest of the algorithms, because it is faster.

Algorithms with the inclusion of a third Phase outperform HGATJ+NSGA2, and the version with β-BRSM is slightly worse. There are no significant differences between the two algorithms with PR. These results indicate that combining PR with an evolutionary algorithm is a good option to solve multiobjective project scheduling problems.

Finally, we are going to compare our algorithms with the best procedure in Ballestín and Blanco (2009), DJGA+NSGA2. We impose different limits in the

number of schedules calculated. Each cell contains the distance to the reference set.

	5000	10000	25000	50000
DJGA+NSGA2	0.2603	0.2168	0.1673	0.1350
HGATJ+NSGA2	0.1679	0.1322	0.0979	0.0794
MOHGATJ (PR I)	0.1527	0.1180	0.0850	0.0670

Table 5: MOHGATJ versus NSGA2

Different conclusions can be drawn from these figures. First of all, MOHGATJ is a good (multi-objective) algorithm, in the sense that it is capable to obtain better results as the schedule limit increases. Secondly, it clearly outperforms DJGA+NSGA2 in every schedule limit. This means that the good performance of HGATJ in the two uniobjective problems has been transferred to the multiobjective tardiness-makespan RCPSP. Finally, the third Phase is useful no matter what the time limit. The relative improvement over HGATJ+NSGA2 is 10%, 12%, 15 and 18.5%.

5.4. Approximation of Pareto sets throughout the algorithm

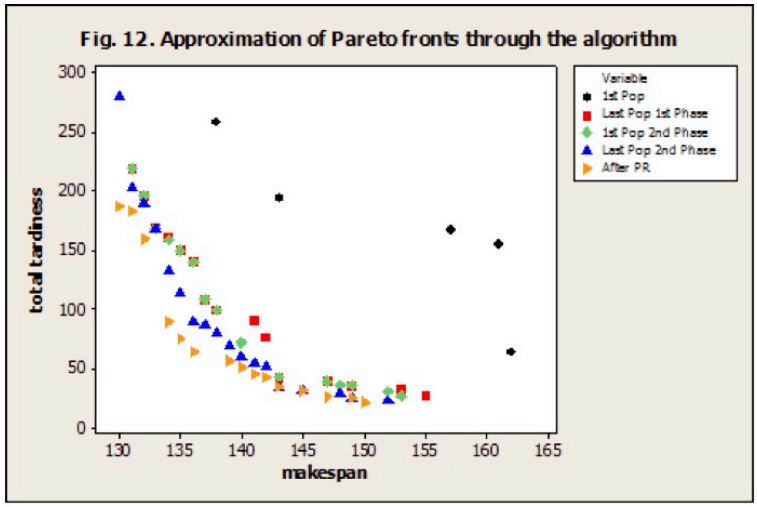

Figure 12: Approximation of Pareto fronts through the algorithm

Figure 12 shows the evolution of the approximation of the Pareto fronts through a version of MOHGATJ in a specific instance, j12013_1.sm with loose due dates. There are 5 approximation sets depicted: a) the first population of the algorithm, before beginning the evolutionary algorithm, b) the last population

obtained after the first phase, c) the first population of the second phase, d) the last population obtained after the second phase, before applying PR and e) the outcome of the algorithm, the approximation obtained after applying PR.

This graph contains some of the trends that happen in many instances. First of all, the greatest improvement comes from the first phase. Secondly, the construction of the population of the second phase does not improve much the Pareto solution already calculated. Nevertheless, sometimes it obtains new efficient solutions between two old efficient solutions or near one old efficient solution. In our case, (148, 36) between (147, 39) and (149, 35). Furthermore, the second phase obtains a significative improvement in the Pareto front, although very far away from the improvement attained in the first phase. Finally, the PR is capable to refine the Pareto front. In this instance an almost complete new front is obtained. In other instances only some of the existing solutions are improved. Also new solutions between two old efficient solutions can be discovered, as in the construction of the second population.

6. Summary and conclusions

There are very few algorithms for project scheduling with due dates, despite their tremendous importance in the field. In this paper we have taken HGA, a very good algorithm for the basic RCPSP, and transformed it into an algorithm for two problems concerning due dates and project scheduling, the TardinessRCPSP and the DeadlineRCPSP. The most important contribution concerns the development of new forms of justification that exploit the information given by the due dates. The performance of these types of justification depends on the instance. We have added an extra gene in HGA creating HGATJ, a self-adapting evolutionary algorithm. This gene handles the use of the justification in the algorithm.

In the second part of the paper we have adapted HGATJ to solve the multiobjective tardiness-makespan RCPSP, creating MOHGATJ. A third phase with PR has been added to MOHGATJ, improving its performance. Both HGATJ and MOHGATJ clearly outperform the few algorithms from the literature that exist for the corresponding problems.

References

[1] Ballestín F., Valls V. and Quintanilla S. (2006). Due Dates and RCPSP. In: *Józefowska, J., and Weglarz, J. (eds.): Perspectives in Modern Project Scheduling,* Springer, Berlin.

[2] Ballestín F., Valls V., and Quintanilla M. (2009). Scheduling projects with limited number of preemptions. *Comput. Oper. Res.*, **36**, 2913 - 2925.

[3] Ballestín F., and Blanco R. (2011). Theoretical and practical fundaments for multi-objective optimisation in resource-constrained project scheduling problems. *Comput. Oper. Res.*, **38**, 51-62.

[4] Barrios A., Ballestín F., and Valls V. (2011). A double genetic algorithm for the MRCPSP/max. *Comput. Oper. Res.*, **38 (1)**, 33-43.

[5] Brucker P., Drexl A., Möhring R., Neumann K., and Pesch E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, **112**, 3-41.

[6] Czyzak P., and Jaszkiewicz A. (1998). Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization. *J. Multi-Crit. Decis. Anal.*, **7**, 34-47.

[7] Deb K., Agrawal S., Pratap A., and Meyarivan T. (2000). A Fast Elitist Nondominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In: *Schoenauer M., et al. (eds.): Parallel Problem Solving from Nature (PPSN VI),* Springer, Berlin.

[8] Dixon, W.J., and Mood, A.M. (1946). The statistical sign test. *J. Am. Statist. Assoc.*, **41**, 557–566.

[9] Drezet, L.E., and Billaut, J.C. (2008). A project scheduling problem with labour constraints and time-dependent activities requirements. *Int. J. Prod. Econ.*, **112**, 217–225

[10] Fisher R.A. (1925). *Statistical Methods for Research Workers*, Oliver and Boyd, Edinburgh.

[11] Garey M.R., and Johnson D.S. (1979). *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco.

[12] Glover F., and Laguna M. (1999). Fundamentals of scatter search and path relinking. *Control Cybern.*, **29**, 653-684.

[13] Gordon V., Proth J-M., and Chu C. (2002). A survey of the state-of-the art of common due date assignment and scheduling research. *Eur. J. Oper. Res.*, **139**, 1-25.

[14] Hartmann S., and Kolisch R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *Eur. J. Oper. Res.*, **127**, 394-407.

[15] Herroelen W., Demeulemeester E., and De Reyck B. (1998). Resource-constrained project scheduling: a survey of recent developments. *Comput. Oper. Res.*, **25**, 279-302.

[16] Herroelen W., Demeulemeester E., and De Reyck B. (1998). A Classification Scheme for Project Scheduling. In: *Weglarz J. (ed.): Handbook on Recent Advances in Project Scheduling*, Kluwer Academic Publishers.

[17] Józefowska J., and Weglarz J. (eds.). (2006). *Perspectives in modern project scheduling*, Springer, Berlin.

[18] Kis T. (2005). A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. *Math. Program.*, **103 (3)**, 515-539.

[19] Klein R., and Scholl A. (2000). Scattered branch and bound - An adaptive search strategy applied to resource-constrained project scheduling. *Cent. Eur. J. Oper. Res.*, **7**, 177-201.

[20] Kolich R. (1995). *Project Scheduling Under Resource Constraints – Efficient Heuristics for Several Problem Classes*, Phisica-Verlag, Heidelberg.

[21] Kolisch R., and Sprecher A. (1997). PSPLIB - A project scheduling library. *Eur. J. Oper. Res.*, **96**, 205-216.

[22] Slowinski R. (1989). Multiobjective project scheduling under multiple-category resource constraints. In: *Slowinski R., and Weglarz J. (eds.): Advances in Project Scheduling*, Elsevier, Amsterdam.

[23] Özdamar L., and Ulusoy G. (1995). A survey on the resource-constrained project scheduling problem. *AIIE Transactions*, **27**, 574-586.

[24] Valls V., Ballestín F., and Quintanilla S. (2007). A Hybrid Genetic Algorithm for the Rcpsp. *Eur. J. Oper. Res.*, **185**, 495–508.

[25] Valls V., Ballestín F., and Quintanilla S. (2005). Justification and RCPSP: a technique that pays. *Eur. J. Oper. Res.*, **165**, 375-386.

[26] Valls V., Ballestín F. and Quintanilla S. (2006). Justification technique generalisations. In: *Józefowska J., and Weglarz J. (eds.): Perspectives in Modern Project Scheduling*, Springer, Berlin.

[27] Vanhoucke M., Demeulemeester E., and Herroelen W. (2001). Exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem. *Ann. Oper. Res.*, **102**,179-196.

[28] Viana A., and Pinho de Sousa J. (2000). Using metaheuristics in multiobjective resource constrained project scheduling. *Eur. J. Oper. Res.*, **120**, 359-374.

About the authors

Francisco Ballestín es profesor titular en la Universidad de Valencia. Realizó su tesis doctoral en secuenciación de proyectos, donde ha centrado parte de su investigación. También ha trabajado en aplicaciones prácticas para problemas de almacenes y de hospitales. Ha publicado en revistas como Production and Operations Management y European Journal of Operational Research.

Rosa Blanco Gómez obtuvo el título de Ingeniero en Informática por la Universidad del País Vasco en el año 2000 y se doctoró en Ciencias de la Computación en la misma universidad en el 2005. Desde Enero de 2006 forma parte del Departamento de Estadística e Investigación Operativa en la Universidad Pública de Navarra donde imparte docencia y desarrolla labores investigadoras dentro del grupo de investigación DECYL (Datos, Estadística, Calidad y Logística). Su interés investigador ha evolucionado desde los sistemas probabilísticos y las redes Bayesianas a los métodos estadísticos y sistemas de optimización para la simulación de sistemas estocásticos y dinámicos, participando en distintos proyectos de investigación donde se aplica esta investigación a sistemas de energía renovables.