European Journal of Operational Research 199 (2009) 46–54

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Activity list representation for a generalization of the resource-constrained project scheduling problem ☆

Khaled Moumene, Jacques A. Ferland *

Dept. Informatique et Recherche Opérationnelle, Université de Montréal, C.P. 6128, Succursale Centre-Ville, Montréal, Québec, Canada H3C 3J7

ARTICLE INFO

Article history:
Received 26 July 2006
Accepted 28 October 2008
Available online 8 November 2008

Keywords:
Project scheduling
Resource-constrained project scheduling
Generalized resource-constrained project scheduling
Activity list
Activity set list

ABSTRACT

Most of the real life scheduling problems include several constraints in addition to the precedence and resource constraints considered in the resource-constrained project scheduling problem (*RCPSP*). In this paper, we define a generalization of the (*RCPSP*) with a wide class of additional constraints, including (but not limited to): a pair of activities must be separated by at least a given duration; a subset of activities cannot be processed simultaneously; an activity cannot start before a particular period; an activity cannot be scheduled in a particular time window; there are resource constraints with varying required and available quantities. We show that for this generalization the activity list and the activity set list representations can be used as efficiently as in the (*RCPSP*) and that by using these representations the optimal solution can always be reached.

This allows most of the known solution procedures for (*RCPSP*) based on these representations to be extended for the generalized (*RCPSP*) by simply replacing the classical decoding procedure used for the (*RCPSP*) with the generalized version introduced here.

1. Introduction

The resource-constrained project scheduling problem (*RCPSP*) has been studied by several authors. It can be summarized as follows. A project including N activities has to be completed in order to minimize some objective function. The makespan is the most commonly used objective function. Each activity i has a duration specified in terms of a number of periods, and two kinds of constraints are considered. The precedence constraints require each activity i to be scheduled after the completion of all its immediate predecessors included in the set P_i. Furthermore, each activity i requires r_{ik} units of resources $k \in R = \{1, 2, \ldots, K\}$ during each period of its completion. The resource constraints limit the number of units of resources $k \in R = \{1, 2, \ldots, K\}$ available during each period of the horizon.

The (*RCPSP*) is known to be NP-Hard (Blazewicz et al., 1983), which implies that the resolution of large instances with an exact method is very time consuming. Several solution procedures have been proposed in the literature. They can be classified into three categories: exact methods (Demeulemeester and Herroelen, 1992; Mingozi et al., 1998; Patterson et al., 1989) using mainly various branch-and-bound procedures; heuristic methods based on the serial and the parallel schedule generation schemes (Boctor, 1990; Demeulemeester and Harroelen, 1995; Kolisch and Drexl, 1996; Kolisch, 1996b); finally, metaheuristic methods based on tabu search (Baar et al., 1998; Nonobe and Ibaraki, 2002; Pinson et al., 1994), simulated annealing (Boctor, 1996; Bouleimen and Lecocq, 2003; Cho and Kim, 1997) and genetic algorithms (Alcaraz and Maroto, 2001; Alcaraz et al., 2004; Hartmann, 1998; Kohlmorgen et al., 1999; Mendes et al., 2009; Valls et al., 2003, 2008). Surveys on several solution procedures can be found in Brucker et al. (1999), Demeulemeester and Herroelen (2002), Hartmann and Kolisch (2000), Herroelen et al. (1998), Kolisch and Hartmann (2006), Kolisch and Hartmann (1999) and Kolisch and Padman (2001).

The (*RCPSP*) underlies several applications, but in general their models also include additional constraints. Brucker and Knust (2001) mentioned three different timetabling problems that can be formulated as (*RCPSP*) with additional constraints. In high-school timetabling (Schaerf, 1999b), the lectures are the activities to be scheduled and the teachers, the student groups and the classrooms are the resources. The objective is to specify a feasible schedule using a specified number of periods. University course timetabling (Schaerf, 1999a) is quite similar, except that individual student registrations are taken into account. In these problems, we may have additional constraints requiring that some pairs of lectures be scheduled simultaneously, or we may formulate the problems as (*RCPSP*) with multiple modes to account for the fact that the lectures can take place in different types of classrooms. The third timetabling problem mentioned in Brucker and Knust (2001) is the audit-scheduling problem (Brucker and Schumacher, 1999) where the jobs to be audited are the activities and the auditors

☆ This research was supported by NSERC grant (OGP 0008312).
* Corresponding author. Tel.: +1 514 343 5687; fax: +1 514 343 5834.
E-mail addresses: khaled_moumene_75@yahoo.com (K. Moumene), ferland@iro.umontreal.ca (J.A. Ferland).

0377-2217/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2008.10.030

are the resources. In this problem, each job has a release time and a due date, the execution of the jobs can be pre-empted, and the auditors are available in disjoint time intervals. Finally, there may be a mismatching cost c_{ij} if job i is audited by auditor j. The objective function is to minimize the sum of the mismatching costs and the tardiness of the job auditing completions. The problem of training a group of persons to perform a set of tasks in an enterprise can also be formulated as a (*RCPSP*) with additional constraints. Indeed, each person may have to be trained in a specified order for the different tasks, leading to precedence constraints. The limited number of teachers and pieces of equipment for training induces resource constraints. Furthermore, there may exist additional constraints to limit the time delay between the training for some pairs of tasks, or to account for the fact that the training is individual for some tasks and a group session for other tasks. In Cesta and Oddi (2002) and De Reyck and Herroelen (1998) the authors introduce a generalization of the precedence constraints (referred to as (GPR)) where minimal time lags between the ending period and the starting period of different pairs of activities are specified.

The (*PRCPSP*) has been also considered by several authors. In this problem, preemption of the processing of the activities is allowed. See Herroelen et al. (1998) for more details. Other variants of the (*RCPSP*) are summarized in Brucker et al. (1999) and Herroelen et al. (1998).

The purpose of this paper is to see how the activity list representation *AL* can be used for a very large class of generalizations of the (*RCPSP*) since most real scheduling problems are much more complex requiring additional constraints. To the best of our knowledge, no such work has been done in the literature for the (*RCPSP*). Our motivation is prompted by the number of successful use of *AL* to solve the (*RCPSP*) and its various interesting properties. For the (*RCPSP*), an *AL* is quickly decoded into a feasible schedule, its list form allows various operators to be applied, and the set of all schedules induced by *AL* always contains the optimal solution (Kolisch, 1996a; Sprecher et al., 1995).

The paper is organized as follows. In Section 2, we define a generalized version of the (*RCPSP*). We introduce the conditions under which an *AL* representation can be used for the problem according to both the forward and the backward scheduling mode in Section 3. In Section 4 we give a necessary condition for the existence of an *AL* inducing an optimal schedule and we introduce many examples of instances verifying this condition. Furthermore, we indicate several other extensions of techniques based on *AL* and developed initially for the (*RCPSP*). Finally, an extension of another representation (activity set list) reducing the search space of all the *AL* is made in Section 6.

2. Generalized (*RCPSP*)

In this paper we propose a generalized (*RCPSP*) problem denoted $(ORD)+(C)$. The (ORD) block includes an objective function to be minimized and the set of all the precedence constraints between activities, which can be empty. The (C) block is a set of additional arbitrary constraints. This set can also be empty.

Accordingly, the (*RCPSP*) is an $(ORD)+(C)$ problem where the objective function of (ORD) is the total makespan and (C) includes the resource constraints. Also, the classical scheduling problem which can be solved optimally with the CPM method is an $(ORD)+(C)$ problem where the objective function of (ORD) is the total makespan and where (C) is empty.

3. Activity list representation for the generalized (*RCPSP*)

The activity list representation (*AL*) of a solution (schedule) for the (*RCPSP*) is a permutation vector of the activities satisfying the precedence constraints. Hence, each activity is positioned in the list after all its predecessors. To obtain the corresponding schedule, the *AL* is decoded with the serial SGS method proposed by Kelley (1963) where the activities are selected according to their order in the list and scheduled at their earliest start period. See also Kolisch (1996a) for more details.

The *AL* representation is used extensively to solve the (*RCPSP*) (Alcaraz and Maroto, 2001; Bouleimen and Lecocq, 2003; Fleszar and Hindi, 2004; Hartmann, 1998; Hindi et al., 2002; Nonobe and Ibaraki, 2002) because it is easily and rapidly decoded, it always induces a feasible solution, its list form is easily manipulated, and there always exists an *AL* inducing an optimal schedule (Kolisch, 1996a; Sprecher et al., 1995). We propose to extend the use of *AL* to $(ORD)+(C)$ problems. But first we have to specify conditions allowing a feasible schedule to be constructed for any *AL* and conditions that guarantee the existence of an *AL* generating the optimal solution.

For the $(ORD)+(C)$ problem, if there are precedence constraints between activities in the (ORD) block, then only these constraints are accounted for in the *AL* representation since each activity is positioned after all predecessors. Such an *AL* is said to be a valid *AL*.

To decode a valid *AL*, a method similar to the serial SGS is used. The activities are selected in their order in *AL*, and they are scheduled at their earliest starting period after the completion of all its predecessors such that all the constraints defined in $(ORD)+(C)$ are satisfied.

Now additional conditions need to be imposed on the (C) block constraints in order to make sure that the decoding scheme is working. To illustrate that, consider this simple example where two activities, 1 and 2, of duration 2 have to be scheduled. Furthermore, additional constraints require that these activities cannot be scheduled at the same time and activity 1 cannot start after period 2. This is an $(ORD)+(C)$ problem where there are no precedence constraints and where the (C) block constraints are specified as above. For this instance, there are only two valid *AL*: $a_1=[1,2]$ and $a_2=[2,1]$. On one hand, a_1 can be decoded into a schedule where activities 1 and 2 start at periods 1 and 3, respectively. On the other hand, a_2 cannot be decoded. Indeed, once activity 2 is first scheduled to start in period 1, then the first additional constraint would induce activity 1 to start in period 3, contradicting the second additional constraint.

Next, we introduce a simple condition on (C) to avoid such a situation.

3.1. Flexible constraints

Definition 1. Let a be any valid *AL* for an $(ORD)+(C)$ problem. A specific constraint $c\in(C)$ is *flexible* if for any activity i and any partial schedule of the activities positioned before i in a, i can be scheduled, accounting only for constraint c, at some period $Di(c)$ or any period later.

Such a constraint is denoted c^f.

Definition 2. The (C) block is *flexible* if all $c\in(C)$ block are c^f.

Such a block is denoted a (C^f) block.

It follows that for any valid *AL* for an $(ORD)+(C^f)$ problem, any activity i in *AL* can be scheduled at some period $D_i(C)=Max_{c\in(C)}\{D_i(c)\}$ or at any period later. It is easy to verify that any valid *AL* for an $(ORD)+(C^f)$ problem can be decoded into a feasible schedule using the serial SGS described before. Note also that the resource constraints in the (*RCPSP*) are flexible, and hence, it is an $(ORD)+(C^f)$ problem.

Furthermore, it follows from Definition 2, that adding any number of flexible constraints to any (*RCPSP*) generates an $(ORD)+(C^f)$ problem. Here are examples of *flexible constraints*:

1. An activity cannot start *before* a particular period.
2. A pair of activities must be separated by *at least* a duration *DR*.
3. An activity *cannot be scheduled in a particular time window*.

But the following constraints are *non-flexible*:

1. An activity *i* cannot start *after* a particular period.
2. Some activities *must be scheduled at the same time*.
3. An activity *must be scheduled* in a particular time window.

To illustrate these notions, we consider the following $(ORD)+(C^f)$ problem where flexible constraints are added to $(RCPSP)$. The problem is illustrated in Fig. 1. One and ten are dummy activities representing the start and end of the schedule, respectively. Six units of a unique resource are available at each period. The additional flexible constraints are the following:

1. The activities 2, 4 and 8 cannot start before periods 2, 7 and 9, respectively.
2. Activities 7 and 8 cannot be processed simultaneously.

The schedule associated with

$AL_1 = [1,2,3,4,5,6,7,8,9,10]$

is in Fig. 2.

- Activity 2 is scheduled at period 2 because it cannot start before this period.
- Activity 4 is scheduled at period 7 because:
  - it has activity 2 as a predecessor (cannot be scheduled before period 6).
  - it cannot start before period 7.
- Activity 6 is scheduled at period 8 because:
  - it has activities 2 and 3 as predecessors (cannot be scheduled before period 6).
  - resources are insufficient until period 8.
- Activity 8 is scheduled at period 13 because:
  - it has activity 6 as a predecessor (cannot be scheduled before period 11).
  - it cannot start before period 9.
  - resources are insufficient until period 11.
  - it cannot be processed simultaneously with activity 7 (cannot be processed in periods 9, 10, 11 and 12).

3.2. The backward mode

So far, we have generalized the use of *AL* for the $(ORD)+(C^f)$ problem with respect to the forward decoding mode where the activities are scheduled sequentially from the first activity to the last in the AL at their earliest starting period after the completion of their predecessors. But activities can also be scheduled using

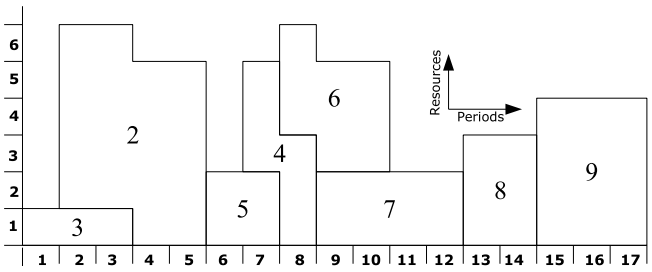

Fig. 2. Schedule associated with AL_1.

the backward mode. This idea was introduced by Li and Willis (1992) and Ozdamar and Ulusoy (1996). Activities are scheduled sequentially from the last activity to the first in the *AL* at their latest starting period in order to be completed before their successors.

Activities can be scheduled starting from an upper bound *T*. The dummy end activity is scheduled first to end at period *T* and the other activities are scheduled as late as possible without violating the precedence and the resource constraints. The starting times of the activities are then reduced by the same duration such that the dummy start activity starts at the first period (Klein, 2000).

Various promising heuristics have been proposed for the $(RCPSP)$ using both modes (Alcaraz and Maroto, 2001; Alcaraz et al., 2004; Klein, 2000). In this section we discuss the conditions allowing any valid *AL* for the $(ORD)+(C)$ problem to be decoded with the backward mode.

Definitions 1 and 2 can be adapted for the backward mode as follows.

Definition 3. Let *a* be any valid *AL* for a $(ORD)+(C)$ problem. A specific constraint $c \in (C)$ is *backward flexible* if for any activity *i* and any partial schedule of the activities positioned after *i* in *a*, *i* can be scheduled, accounting only for constraint *c*, at some period $D_i^B(c)$ or any period sooner.

Such a constraint is denoted c^{bf}.

Definition 4. The (C) block is *backward flexible* if all $c \in (C)$ block are c^{bf}.

Such a block is denoted a (C^{bf}) block.

The following result shows that flexibility and backward flexibility are *equivalent*.

Theorem 1. *The* (C) *block is flexible if and only if it is backward flexible.*

Proof 1. First, we show that if the (C) block is flexible *then* it is backward flexible. For the sake of contradiction, suppose that the (C) block is flexible but not backward flexible. Hence, there is an *AL a* and a *first* activity *i* in *a* that cannot be scheduled with the backward mode at a period D_i^B or at any period sooner. Let:

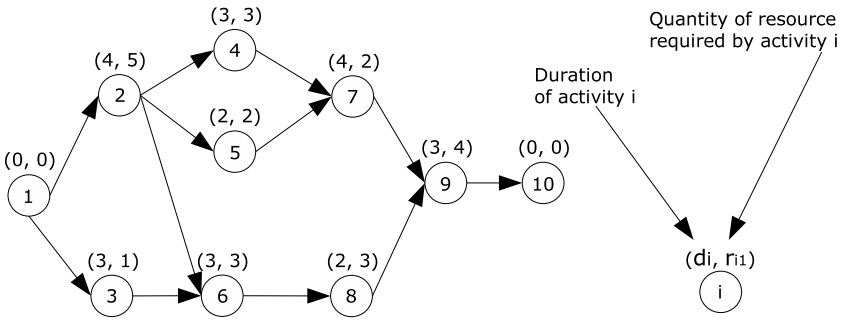

Fig. 1. Example of an $(ORD)+(C^f)$ problem.

- A^i_{bf}, A^i_{af}: the sets of activities positioned in a before and after i, respectively.
- $P^B_{A^i_{af}}$: the partial schedule obtained by scheduling the activities A^i_{af} with the backward mode.
- P^F: the schedule obtained by decoding a with the forward mode.
- DR^B_j: the duration between the starting period of an activity $j \in A^i_{af}$ and the starting period of the activity that starts first in $P^B_{A^i_{af}}$.

Now, add new flexible constraints $(\overline{C})$ to the problem $(ORD)+(C)$, and denote $(ORD)+(C)+(\overline{C})$ the resulting problem. To specify the constraints $(\overline{C})$, recall that $D_j(C)$ denote the period where activity j can be scheduled or any subsequent period according to the forward mode. Now, for each activity $j \in A^i_{af}$, there is a constraint in $(\overline{C})$ requiring that the activities i and j must be at least $D_{max}+DR^B_j$ periods apart (i.e., the number of periods between the ending period of i and the starting period of j is at least equal to $D_{max}+DR^B_j$). Furthermore, D_{max} is selected large enough such that when decoding the AL a with the forward mode for the problem $(ORD)+(C)+(\overline{C})$, each activity $j \in A^i_{af}$ is scheduled at period $D_j(C)$ or later.

Now, consider the feasible schedule for the problem $(ORD)+(C)+(\overline{C})$ obtained by decoding a using the forward mode. Activity i and each activity $j\prime \in A^i_{bf}$ are scheduled to start at the same period as in P^F. Each activity $j \in A^i_{af}$ is then selected to start at period $D_{max}+DR^B_j$ since D_{max} is selected such that $Max\{D_j(C), D_{max}+DR^B_j\} = D_{max}+DR^B_j$. Furthermore, these starting periods for the activities $j \in A^i_{af}$ verify the precedence constraints since they are obtained by shifting those in $P^B_{A^i_{af}}$ by D_{max} periods (see Fig. 3 for an illustrative example).

Next, consider AL a for the problem $(ORD)+(C)+(\overline{C})$ to be decoded using the backward mode. Then activities in AL are scheduled as in $P^B_{A^i_{af}}$ since the constraints in $(\overline{C})$ are not relevant in scheduling these activities. Furthermore, since these activities can be scheduled $D_{max}+DR^B_j$ periods after activity i when AL a is decoded with the forward mode, it follows that i can be scheduled at least $D_{max}+DR^B_j$ periods before any activity $j \in A^i_{af}$ using the backward mode. Hence there exists a period D^B_i such that activity i can be scheduled at this period or sooner (using a value of D_{max} large enough). Since the problem $(ORD)+(C)+(\overline{C})$ has more constraints, it follows that the same result is true for the problem $(ORD)+(C)$. Thus, the (C) block is backward flexible.

A similar argument can be used to show that if the (C) block is backward flexible, then it is flexible. □

The following result follows immediately.

Corollary 1. *All valid AL can be decoded into feasible schedules for the* $(ORD)+(C^f)$ *problem using the forward or the backward mode.*

To illustrate the backward scheduling mode, we can use the example in Fig. 1. The schedule associated with $AL_1 = [1,2,3,4,5,6,7,8,9,10]$ obtained with the backward mode is illustrated in Fig. 4.

In the rest of the paper we use only the notion of flexibility since the results are also valid for backward flexibility.

4. Optimal activity list for $(ORD)+(C^f)$

As mentioned before, if the $(RCPSP)$ has an optimal schedule, there always exists a valid AL associated with an optimal schedule (Kolisch, 1996a; Sprecher et al., 1995). Next, we discuss this issue for the $(ORD)+(C^f)$ problem.

First, we need to introduce some useful definitions. Some of these definitions were introduced by Backer (1974) for the job shop problem and then extended and formalized by Sprecher et al. (1995) for the $(RCPSP)$. Now, we extend these to the $(ORD)+(C^f)$ problem.

Definition 5. A *left shift* of activity j in a feasible schedule consists in scheduling j sooner without changing the scheduling period of the other activities. The resulting schedule must be feasible.

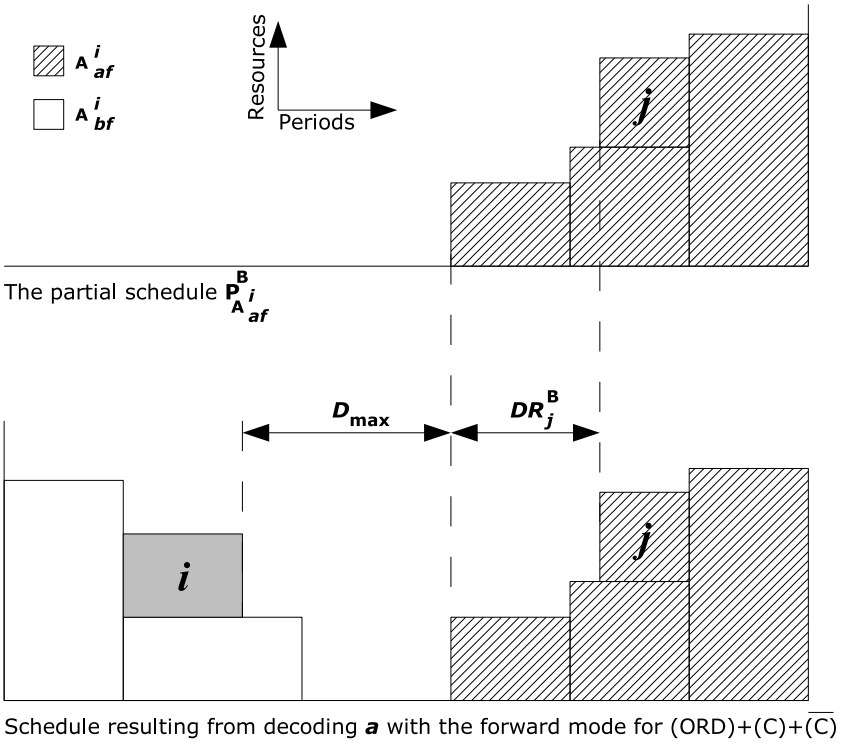

Fig. 3. Illustrative example.

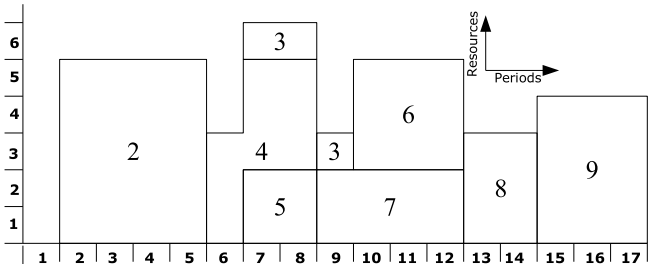

Fig. 4. Schedule example obtained with the backward mode.

Definition 6. A *one-period left shift* of an activity j is a left shift of j where j is scheduled one period sooner.

Definition 7. A *local left shift* of activity j is a left shift of j obtained by applying successively one or more one-period left shifts.

Definition 8. A *global left shift* of activity j is a left shift of j that cannot be obtained only by applying successively one or more one-period left shifts.

Definition 9. A *semi-active* schedule is a feasible schedule where each activity cannot be locally left shifted.

Definition 10. An *active* schedule is a feasible schedule where each activity cannot be locally or globally left shifted.

It is easy to verify that *any valid AL induces an active schedule* for $(ORD) + (C^f)$ since each activity is scheduled at its earliest start period, hence it cannot be scheduled sooner without rescheduling other activities.

Definition 11. For a minimization scheduling problem, the objective function is *regular* if it is non-decreasing with the completion times of the activities.

Note that the *makespan*, a frequently used objective function for scheduling problems like $(RCPSP)$, *is a regular objective function*. Many other regular objective functions can be found in Patterson et al. (1990) and Slowinski (1989) for scheduling problems.

Definition 12. Let P be a feasible schedule, i an activity, and AP_i the set of activities starting at the same period as i or later. A *partial schedule* P_i *of* P is the schedule for the activities $AP_i \cup \{i\}$. These activities are scheduled at the same periods as in P.

Definition 13. Let i be an activity, P_i a partial schedule of P, and Ps_i the other activities than AP_i scheduled at the same period as in P.

- $C\bar{O}N_i$ is a subset of the constraints of $(ORD) + (C^f)$. Activities concerned by each of these constraints are in Ps_i *only*.
- CON_i is a subset of the constraints of $(ORD) + (C^f)$. At least one activity in P_i is concerned by each of these constraints.

Definition 14. Let i be an activity and d_i its starting period in the partial schedule P_i. P_i is *locally extensible* if a local left shift of i in P_i of $(d_i - 1)$ periods is possible considering *only* constraints CON_i.

According to Definition 14, P_i is locally extensible if activity i can be left shifted to any period before d_i when only constraints CON_i apply.

Definition 15. A feasible schedule P is *globally extensible* if, for each activity i, the partial schedule P_i of P is locally extensible.

In the following sections, we discuss the existence of an optimal *AL* for the problem $(ORD) + (C^f)$ *having a regular objective function*. This problem is denoted $(ORD^r) + (C^f)$.

4.1. The flexibility condition

Relying on the results in Section 3, the flexibility condition allows any valid *AL* to be decoded into a feasible schedule for the problem $(ORD) + (C^f)$. Here, we illustrate that this condition is not sufficient in general for the existence of an optimal valid *AL*.

Consider the following simple $(ORD^r) + (C^f)$ example. It is an $(RCPSP)$ where other flexible constraints are added. Fig. 5 shows the principal data of the $(RCPSP)$. Only one resource is used, and 4 units are available during each period.

The additional flexible constraints are:

1. Activities 1 and 2 cannot be separated by exactly 2 periods.
2. Activities 2 and 3 cannot be separated by exactly 2 periods.
3. Activity 3 cannot start before period 7.

For this example, there exist *only* 2 *valid AL*, $a_1 = [1,2,3,4]$ and $a_2 = [1,2,4,3]$. a_1 and a_2 induce the same schedule illustrated in Fig. 6 when they are decoded using the forward scheduling mode. This schedule has a duration of 9 periods.

Now, consider the following feasible schedule in Fig. 7. This schedule has a duration 8, which is a better makespan then the one induced by a_1 and a_2.

This example illustrates clearly that *the set of schedules induced by all valid AL does not contain the optimal schedule*. Hence, the flexibility condition is not sufficient to guarantee the existence of an optimal valid *AL*.

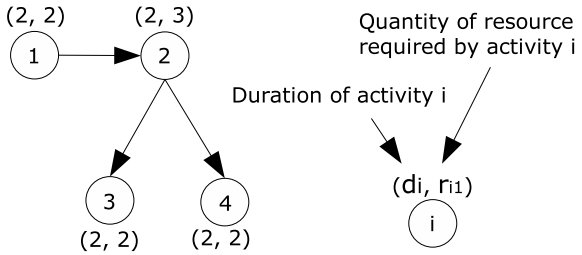

Fig. 5. Example of an $(ORD^r) + (C^f)$ problem.

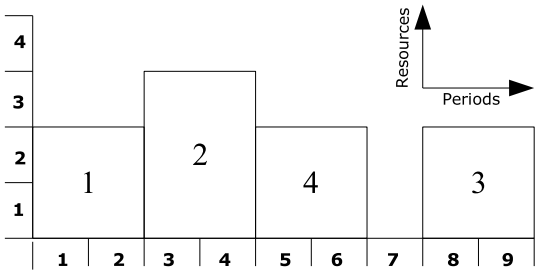

Fig. 6. Schedule resulting from a_1 and a_2.

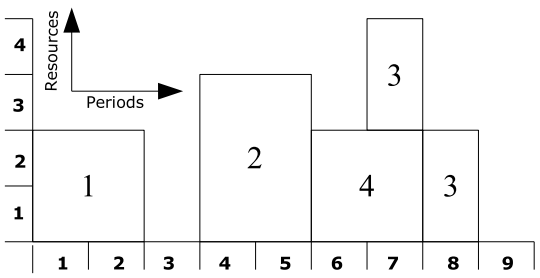

Fig. 7. Example of a better schedule.

This example also illustrates that the set of schedules induced by all the valid *AL may not contain all active schedules* because the schedules of Figs. 6 and 7 are both active.

4.2. Sufficient conditions

In this section we give a new condition to guarantee the existence of an optimal valid *AL* for the $(ORD^r) + (C^f)$ problem. We use the following two important results.

Theorem 2. *If a $(ORD^r) + (C^f)$ problem has an optimal schedule then it has an optimal active schedule.*

Proof 2. Let a_{opt} be an optimal schedule. If a_{opt} is active then the result is true. Otherwise, it is possible to apply a left shift to a_{opt}. After this operation, the value of the objective function for the new schedule remains unchanged since it is regular and since a_{opt} is optimal. Successive left shifts are then applied as long as possible. The number of left shifts applied is finite because each time a left shift is applied, an activity is scheduled at least one period earlier and the number of periods is finite. Since a left shift cannot be applied on the resulting schedule, then it is active. This schedule has the same objective function value as a_{opt}, and it is an optimal active schedule. □

Note that the fact that the (C^f) block is flexible is not used explicitly in the proof. However, this condition is required to ensure that any valid *AL* can be decoded into a feasible schedule.

The second result introduces a condition guaranteeing the existence of a valid *AL* associated with each active schedule.

Theorem 3. *Consider the $(ORD) + (C^f)$ problem. If all the feasible schedules are globally extensible then there exists a valid AL associated with each active schedule.*

Proof 3. Let P be a feasible active schedule, and denote by a_p an *AL* constructed according to the following process. At step n of the process, the activity in position n of a_p is selected. First we determine the set of candidate activities having all their predecessors already selected. An earliest starting period SP_j is determined for each candidate activity j according to the constraints of the problem and accounting for the subset of activities already selected. The activity in position n of a_p is selected among the candidate activities having its earliest starting period identical to its starting period in P.

Clearly, if all the activities are selected sequentially in this process, then the valid *AL* constructed is associated with the active schedule P. To complete the proof, suppose for the sake of contradiction that at some step n_0 of the process, for each candidate activity j, SP_j is different from its starting period SP_j^P in P.

For some candidate activity j, suppose that $SP_j^P < SP_j$. By the definition of SP_j, activity j cannot be scheduled earlier than period SP_j according to the constraints of the problem. Then activity j cannot be scheduled at $SP_j^P (< SP_j)$ in P since P is feasible by assumption. Hence it is not possible to have $SP_j^P < SP_j$, and therefore $SP_j < SP_j^P$ for all candidate activities at step n_0.

For any candidate activity j, it is clear that the constraints preventing j from being scheduled at period SP_j in P are constraints concerning j and activities that are not selected yet, since they force the starting period of j to be SP_j^P in P. Now denote by i any of the earliest starting candidate activities in P (see Fig. 8). Hence, there are constraints in CON_i that are preventing i from being scheduled at period SP_i in P while constraints $C\bar{O}N_i$ allow it. Furthermore, note that all the activities that are not selected yet appear in P_i since i is any of the earliest starting candidate activities in P and the rest of the activities are direct or indirect successors of the candidate activities.

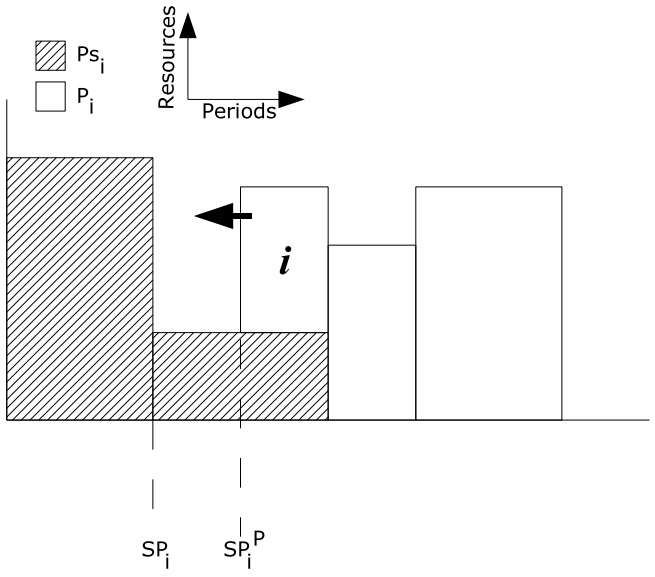

Fig. 8. Example of a schedule with the assumption that i cannot start at period SP_i.

Now, by assumption, P is globally extensible, Hence, activity i can be scheduled at an earlier period in P_i using a local left shift of duration $(SP_i^P - 1)$ with regard to the constraints CON_i. It follows that these constraints are not preventing i from being scheduled at period SP_i rather than at period SP_i^P (by left shifting i of $(SP_i^P - SP_i)$ periods). Thus neither CON_i nor $C\bar{O}N_i$ prevent i from being scheduled at SP_i in P. This contradicts the fact that P is an active schedule.

Hence, it is always possible to construct a valid *AL* associated with any active schedule. □

The following important result is a direct consequence of Theorems 2 and 3.

Corollary 2. *Consider an $(ORD^r) + (C^f)$ problem having an optimal schedule. If all feasible schedules are globally extensible then there exists a valid AL inducing an optimal schedule.*

According to Theorem 2, if the problem has an optimal schedule, then it has an optimal active schedule. Referring to Theorem 3, there exists a valid *AL* associated with the optimal active schedule.

Now, consider two problems $(ORD^r) + (C_1^f)$ and $(ORD^r) + (C_2^f)$ having optimal schedules. Assume also that for each of these problems, each feasible schedule is globally extensible. Let $C_{1,2}^f$ denote the union of the constraint sets (C_1^f) and (C_2^f). It is easy to verify that for the problem $(ORD^r) + (C_{1,2}^f)$, *each feasible schedule is globally extensible*. Indeed, if each partial feasible schedule P_i for $(ORD^r) + (C_{1,2}^f)$ is locally extensible considering constraints (C_1^f) (respectively constraints (C_2^f)), it is locally extensible considering constraints $(C_{1,2}^f)$. This means that if a set is composed of constraints having this property individually, this set also has this property. Since the flexibility is also preserved, *the resulting problem has an optimal valid AL* according to Theorem 2.

It is interesting to note that for the particular case of the $(RCPSP)$, each partial feasible schedule is globally extensible. Since the resource constraints are also flexible, then the problem has an optimal valid *AL*. This result was already introduced in Kolisch (1996a) and Sprecher et al. (1995).

For the example of Fig. 2, the flexible constraints "activities 1 (respectively 2) and 2 (respectively 3) cannot be separated by exactly 2 periods" imply that at least one partial schedule is not locally extensible. For instance, the partial schedule P_2 in Fig. 7 is not locally extensible.

4.3. Example of constraints inducing an optimal AL

As underlined above, the existence of an optimal valid *AL* for $(ORD^r)+(C^f)$ can be proved by (1) showing that *each individual constraint* (c) in (C^f) is flexible and (2) for the problem $(ORD^r)+(c)$, each feasible schedule is globally extensible. We give here a non-exhaustive list of constraints complying with these conditions.

1. A pair of activities must be separated by *at least* a duration *DR*.
2. A subset of activities *cannot be processed simultaneously*.
3. An activity cannot start *before* a particular period.
4. An activity *cannot be scheduled in a particular time window*.
5. Resource constraints similar to those of the (*RCPSP*).
6. Resource constraints similar to those of the (*RCPSP*) but where *quantities of resources are not necessarily identical at each period*. In this case, the availabilities must allow each individual activity to be scheduled at each period.
7. Resource constraints similar to those of the (*RCPSP*) but where *the quantities required by an activity change during its duration*. For instance, an activity of duration 2 may require 2 units of some resource in its first period and 3 units in its second period of its completion. Also in this case, it is important that each individual activity *can be scheduled at each period*.

5. Other (*RCPSP*) extensions

Many solution procedures developed for the (*RCPSP*) use sets of valid *AL*; for examples, Alcaraz and Maroto (2001), Hartmann (1998) and Hindi et al. (2002) (genetic algorithm), Bouleimen and Lecocq (2003) (simulated annealing) and Nonobe and Ibaraki (2002) (tabu search). Most of these procedures can be easily extended to solve the $(ORD)+(C^f)$ problem by simply *adjusting the way that AL are decoded to account for* $(ORD)+(C^f)$ constraints.

Valls et al. (2005) introduce the *justification* operation on a feasible schedule *P* for the (*RCPSP*). Roughly speaking, the *right justification* (respectively *left justification*) of *P* consists of ordering the activities in decreasing (respectively increasing) order by their end period (respectively start period). Then the new feasible schedule P^R (respectively P^L) is generated by scheduling the activities in that order at their earliest ending (respectively latest starting) period. P^R (P^L) has at least as good a makespan as *P*. Generally, a *double justification* is applied to a schedule *P* to obtain the feasible schedule $(P^R)^L$. Other similar operations are summarized in Tormos and Lova (2001). These operations can also be easily extended to the $(ORD^r)+(C^f)$ problem where the makespan is the objective function, by considering the constraints of the problem when activities are rescheduled.

Klein (2000) introduces the *bidirectional planning* for the (*RCPSP*). This method generates a schedule by using the forward and the backward modes simultaneously. Each activity is scheduled according to one mode and the resulting two partial schedules are merged to obtain a final feasible schedule. Since the forward and the backward scheduling mode have been defined for the $(ORD)+(C^f)$ (sections 2 and 3.2) this method can also be extended to this problem.

6. Activity set list for $(ORD)+(C^f)$

The notion of an activity set list *ASL* was introduced by Moumene and Ferland (2005, 2008) as a representation for schedules of the (*RCPSP*). This representation may considerably reduce the search in the space of all *AL* allowing to *avoid selecting several different AL corresponding to the same schedule*. Indeed, an *ASL* may correspond to several different valid *AL* inducing the *same unique* schedule.

An *ASL* is an ordered list of different subsets of activities corresponding to a partition of the set of activities. For each of these subsets, there may exist several permutations where each activity is positioned after its predecessors belonging to the subset. We refer to these permutations as *orders* of the subset.

Furthermore, an *ASL* has the permutation property (*PRMT*) if all *AL* obtained by merging orders of the subsets (one for each subset) are *valid* and correspond to *the same unique schedule*.

For instance, consider the (*RCPSP*) given in Fig. 1 (without the additional constraints). The schedule corresponding to the *ASL*=[{1, 2, 3, 4, 5}{6, 7, 8, 9, 10}] is illustrated in Fig. 9. It is easy to verify that this *ASL* is (*PRMT*) since all the *AL* constructed by merging any order of $sb_1=\{1,2,3,4,5\}$ with one of $sb_2=\{6,7,8,9,10\}$ are valid and correspond to the schedule of Fig. 9. For instance, $o_1=[1,2,5,4,3]$ (respectively $o_2=[6,8,7,9,10]$) is an order of sb_1 (respectively sb_2). Merging these orders induces the valid *AL*

$$[1,2,5,4,3,6,8,7,9,10]$$

that corresponds to the schedule of Fig. 9.

Moumene and Ferland (2005, 2008) introduce a polynomial algorithm (ConstructASLPRMT) to construct an *ASL* (*PRMT*) starting from a valid *AL*. The constructed *ASL* induces the same schedule as the starting *AL*. The algorithm selects activities according to their order in the *AL*. For each selected activity *i*, two periods are calculated: $TP(i)$ the earliest starting period of *i* considering only the precedence constraints, and $TPR(i)$ the earliest starting period of *i* considering the precedence and the resource constraints. If $TP(i)=TPR(i)$, then *i* is included in the current subset. Otherwise, a new subset is initialized with *i*. Then *i* is scheduled to start at period $TPR(i)$ and the next activity is selected. The resulting *ASL* is the list of these subsets ordered according to the order in which they are generated.

Now it is easy to see that the notion of *ASL* (*PRMT*) can be extended to the $(ORD)+(C^f)$ problem. The algorithm above can be extended by defining $TPR(i)$ as the earliest starting period of *i* considering *all the constraints of the problem*.

As an example, consider the problem $(ORD)+(C^f)$ described in Fig. 1. Applying the algorithm to the valid $AL_1=[1,2,3,4,5,6,7,8,9,10]$ induces the following *ASL* (*PRMT*),

$$ASL_1=[\{1,2,3\}\{4,5\}\{6,7\}\{8,9,10\}].$$

Table 1 shows the orders for each of the 4 subsets of ASL_1. Hence, it is possible to construct $2\times2\times2\times1=8$ different valid *AL*, all corresponding to the same schedule given in Fig. 2.

It follows that the *ASL* can also be used for $(ORD)+(C^f)$ to reduce the search in the space of all valid *AL* by *avoiding those that induce the same schedule*. Furthermore, if an optimal valid *AL* exists for $(ORD^r)+(C^f)$, an *optimal ASL* (*PRMT*) *also exists for the problem*.

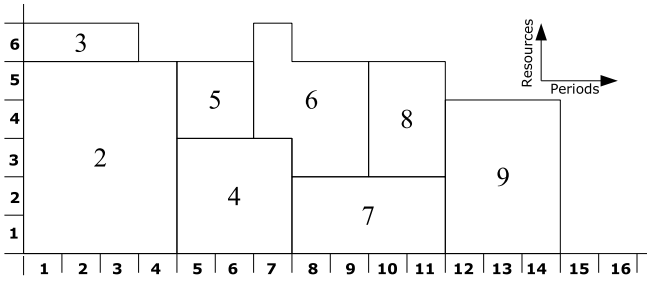

Fig. 9. Example of a schedule induced by an *ASL* and all its corresponding *AL*.

Table 1
Orders corresponding to each subset of ASL_1.

Subsets	Orders
$\{1,2,3\}$	$[1,2,3],[1,3,2]$
$\{4,5\}$	$[4,5],[5,4]$
$\{6,7\}$	$[6,7],[7,6]$
$\{8,9,10\}$	$[8,9,10]$

In fact, it is easy to verify that applying the modified algorithm to an optimal *AL* produces an optimal *ASL* (*PRMT*).

7. Conclusion

We present a wide generalization of the (*RCPSP*) covering several real life applications and an extension of the use of the activity list representation for this new category of problems. This representation has been successfully applied to the (*RCPSP*) within many efficient solution methods due to the fact that an activity list is quickly decoded into a feasible schedule. This representation also lends itself to several operators found in various metaheuristic procedures such as local searches and genetic algorithms.

We extend the decoding procedure of an activity list for our generalization of the (*RCPSP*) in the context of both the forward and the backward modes. Hence, most of the proposed solution procedures for (*RCPSP*) based on this representation can be extended by simply replacing the classic decoding procedure used for the (*RCPSP*) by the generalized version introduced here. We also give sufficient conditions guaranteeing the existence of an optimal schedule corresponding to some activity list. Thus, small instances of some generalized problems can even be solved exactly since, under some conditions, enumerating all possible activity lists leads to some list corresponding to the optimal schedule. Finally, we indicate that the use of the Activity Set List, a recently proposed representation for the (*RCPSP*), can also reduce the activity list search space for this generalization of the (*RCPSP*). Many illustrative examples are also given.

References

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

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

Baar, T., Brucker, P., Knust, S., 1998. Tabu-search algorithms and lower bounds for resource-constrained scheduling problem. In: Voss, S., Martello, S., Osman, I., Roucairol, C. (Eds.), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic, pp. 1–18.

Backer, K.R., 1974. Introduction to Sequencing and Scheduling. Wiley, New York.

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

Boctor, F.F., 1990. Some efficient multi-heuristic procedures for resource-constrained project scheduling. European Journal of Operational Research 49, 3–13.

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

Bouleimen, M., Lecocq, H., 2003. A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple version. European Journal Of Operational Research 149, 268–281.

Brucker, P., Knust, S., 2001. Resource-constrained project scheduling and timetabling. In: Bruke, E., Erben, W. (Eds.), Lecture Notes in Computer Science 2079 PATAT III. Springer, pp. 277–293.

Brucker, P., Schumacher, D., 1999. A new tabu search procedure for audit-scheduling problem. Journal of Scheduling 2, 157–173.

Brucker, P., Drexl, A., Mohring, R., Neumann, K., 1999. Resource-constrained project scheduling: Notation, classification, models and methods. European Journal of Operational Research 112, 3–41.

Cesta, A., Oddi, A., 2002. A constraint-based method for project scheduling with time windows. Journal of Heuristics 8, 109–136.

Cho, J.-H., Kim, Y.-D., 1997. A simulated annealing algorithm for resource-constrained project scheduling problems. Journal of the Operational Research Society 48, 735–744.

Demeulemeester, E., Harroelen, W., 1995. New benchmarking results for the resource constrained project scheduling problem. Management Science 43, 1485–1492.

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

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

De Reyck, B., Herroelen, W., 1998. An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations. Computers and Operations Research 25, 1–17.

Fleszar, K., Hindi, K., 2004. Solving the resource-constrained project scheduling problem by variable neighbourhood search. European Journal of Operational Research 155, 402–413.

Hartmann, S., 1998. A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics 456, 733–750.

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.

Herroelen, W., Reyck, B., Demeulemeester, E., 1998. Resource-constrained project scheduling: A survey of recent developments. Computers and Operations Research 4, 279–302.

Hindi, K.S., Yang, H., Fleszar, K., 2002. An evolutionary algorithm for resource-constrained project scheduling. IEEE Transactions on Evolutionary Computation 6, 512–518.

Kelley, J.E., 1963. The critical-path method: Resources planning and scheduling. In: Muth, J.F., Thompson, G.L. (Eds.), Industrial Planning Scheduling. Prentice-Hall, pp. 347–365.

Klein, R., 2000. Bidirectional planning: Improving priority rule-based heuristics for scheduling resource-constrained projects. European Journal of Operational Research 127, 619–638.

Kohlmorgen, U., Schmeck, H., Haase, K., 1999. Experiences with fine-grained parallel genetic algorithms. Annals of Operations Research 90, 203–219.

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

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

Kolisch, R., Drexl, A., 1996. Adaptative search for solving hard project scheduling problems. Naval Research Logistics 43, 23–40.

Kolisch, R., Hartmann, S., 1999. Heuristic algorithms for solving resource-constrained project scheduling problem: Classification and computation analysis. In: Weglarz, J. (Ed.), Project Scheduling: Recent Models, Algorithms and Applications. Kluwer Academic Publisher, Boston, pp. 147–178.

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

Kolisch, R., Padman, R., 2001. An integrated survey of deterministic project scheduling. OMEGA International Journal of Management Science 29 (3), 249–272.

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

Mendes, J.J.M., Gonćalves, J.F., Resende, M.G.C., 2009. A random key based genetic algorithm for the resource constrained project scheduling problem. Computers and Operations Research 36, 92–109.

Mingozi, A., Maniezzo, V., Ricciardelli, S., Bianco, L., 1998. An exact algorithm for project scheduling with resource constraints based on a new mathematical formulation. Management Science 44, 714–729.

Moumene, K., Ferland, J.A., 2005. Activity Set List Representation for the Resource-Constrained Project Scheduling Problem. Département d'Informatique et de Recherche Opérationnelle, Université de Montréal (Publication No. 1223), March.

Moumene, K., Ferland, J.A., 2008. New representation to reduce the search space for the resource-constrained project scheduling problem. RAIRO – Operations Research 42, 215–228.

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

Ozdamar, L., Ulusoy, G., 1996. A note on interactive forward/backward scheduling technique with a reference to a procedure of Li and Willis. European Journal of Operational Research 89, 400–407.

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: Sowinski, R., Weglarz, J. (Eds.), Advances in project scheduling. Elsevier, Amsterdam, pp. 3–28. 17.

Patterson, J.H., Slowinski, R., Talbot, F.B., Weglarz, J., 1990. Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problem. European Journal of Operational Research 49, 68–79.

Pinson, E., Prins, C., Rullier, F., 1994. Using tabu search for solving the resource-constrained project scheduling problem. In: Proceedings of the Fourth International Workshop on Project Management and Scheduling, Leuven, Belgium, pp. 102–106.

Schaerf, A., 1999a. A survey of automated timetabling. Artificial Intelligence Review 13, 87–127.
Schaerf, A., 1999b. Local search techniques for large high school timetabling problems. IEEE Transactions on Systems, Man and Cybernetics, Part A 29, 368–377.
Slowinski, R., 1989. Multiobjective project scheduling under multiple-category resource consraints. In: Slowinski, R., Weglarz, J. (Eds.), Advances in Project Scheduling. Elsevier, Amsterdam, pp. 151–167.
Sprecher, A., Kolisch, R., Drexel, A., 1995. Semi-active, active and non-delay schedules for resource-constrained project scheduling problem. European Journal of Operational Research 80, 94–102.
Tormos, S., Lova, A., 2001. A competitive heuristic solution technique for resource-constrained project scheduling. Annals of Operations Research 102, 65–81.
Valls, V., Ballestin, F., Quintanilla, M.S., 2003. A hybrid genetic algorithm for the RCPSP. Technical Report, Department of Statistics and Operations Research, University of Valencia.
Valls, V., Ballestin, F., Quintanilla, S., 2005. Justification and RCPSP: A technique that pays. European Journal of Operational Research 165, 375–386.
Valls, V., Ballestín, F., Quintanilla, S., 2008. A hybrid genetic algorithm for the resource-constrained project scheduling problem. European Journal of Operational Research 185, 495–508.