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*: