

Periodically Aggregated Resource-Constrained Project Scheduling Problem

Pierre-Antoine Morin, Christian Artigues, Alain Haït

▶ **To cite this version:**

Pierre-Antoine Morin, Christian Artigues, Alain Haït. Periodically Aggregated Resource-Constrained Project Scheduling Problem. European Journal of Industrial Engineering, 2017, 11 (6), 26p. 10.1504/EJIE.2017.10009144 . hal-01876198

HAL Id: hal-01876198
https://laas.hal.science/hal-01876198

Submitted on 26 Jun 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Open Archive Toulouse Archive Ouverte (OATAO)

OATAO is an open access repository that collects the work of some Toulouse researchers and makes it freely available over the web where possible.

This is an author's version published in: https://oatao.univ-toulouse.fr/19496

Official URL: http://dx.doi.org/10.1504/EJIE.2017.10009144

To cite this version :

Morin, Pierre Antoine and Artigues, Christian and Haït, Alain Periodically Aggregated Resource-Constrained Project Scheduling Problem. (2017) European J. of Industrial Engineering, vol. 11 (n° 6). pp. 792-817. ISSN 1751-5254

Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr

Periodically aggregated resource-constrained project scheduling problem

Pierre-Antoine Morin*

ISAE SUPAERO,
University of Toulouse,
Toulouse, France
and
LAAS CNRS,
University of Toulouse,
CNRS, Toulouse, France
Email: pierre-antoine.morin@isae.fr
*Corresponding author

Christian Artigues

LAAS CNRS,
University of Toulouse,
CNRS, Toulouse, France
Email: artigues@laas.fr

Alain Haït

ISAE SUPAERO,
University of Toulouse,
Toulouse, France
and
LAAS CNRS,
University of Toulouse,
CNRS, Toulouse, France
Email: alain.hait@isae.fr

Abstract: In this paper, a new project scheduling problem is introduced, the periodically aggregated resource-constrained project scheduling problem (PARCPSP), in which the resource usage is considered on average over aggregated periods of parameterised length, while temporal aspects (start/completion dates of activities and precedence relations) are taken into account in an exact manner. A mixed integer linear programming formulation based on a mixed time representation is given. The adaptation of schedule generation schemes developed for standard project scheduling problems is discussed. An iterative solution scheme is described; experiments show that this method permits to find better upper bounds and sometimes enables to reduce the overall computational time. [Received 30 September 2016; Revised 28 January; Revised 5 July 2017; Accepted 16 September 2017]

Keywords: periodically aggregated resource-constrained project scheduling problem; PARCPSP; mixed integer linear programming; MILP; schedule generation schemes; SGS; iterative solution scheme; ISS.

Reference to this paper should be made as follows: Morin, P-A., Artigues, C. and Haït, A. (2017) 'Periodically aggregated resource-constrained project scheduling problem', *European J. Industrial Engineering*, Vol. 11, No. 6, pp.792–817.

Biographical notes: Pierre-Antoine Morin is a PhD student at the Complex Systems Engineering Department (DISC) in ISAE-SUPAERO, and he is also a member of the Operations Research, Combinatorial Optimisation and Constraints (ROC) team in LAAS-CNRS, both members of the University of Toulouse, France. He obtained his MSc in Informatics from Polytech Tours, the Engineer School of the University of Tours, France, in 2015, with a specialisation in Logistics and Transport. His field of research is on project planning and scheduling, mathematical programming, and combinatorial optimisation.

Christian Artigues is a Research Director at the LAAS-CNRS, Toulouse, France, and Head of the Operations Research, Combinatorial Optimisation and Constraints team. He received his PhD from the University Paul Sabatier in 1997. Before entering CNRS, the National Center for Scientific Research as a Full Time Researcher in 2006, he was Assistant Professor at the University of Avignon. He carries out researches in discrete optimisation and more precisely on exact methods for scheduling problems, hybrid constraint programming/integer programming methods, robust and reactive scheduling. He published his work in more than 50 international journals and about 40 international conferences in the fields of operations research, constraint programming and industrial engineering. He serves as an Associate Editor of the *Journal of Scheduling*.

Alain Haït is a Professor at ISAE-SUPAERO, Aerospace Engineering School located in Toulouse, France, and a Head of the Complex Systems Engineering Department. He obtained the BE in Mechanical Engineering from the INSA Toulouse, and a PhD in Robotics from the University Paul Sabatier in 1998. Then he served as an Assistant Professor in Industrial Engineering from 1999 to 2007 at ENSIACET, a chemistry and chemical engineering school. From 2003 to 2005, he was Visiting Researcher in the Department of Mathematics and Industrial Engineering of the Ecole Polytechnique de Montréal, Canada. From 2007 to 2015, he was Head of the Industrial Engineering Section of ISAE-SUPAERO. His research area is planning and scheduling for industrial applications, considering human resources and energy constraints.

This paper is a revised and expanded version of a paper entitled 'A new relaxation of the resource-constrained project scheduling problem' presented at the 15th International Conference on Project Management and Scheduling (PMS 2016), Valencia, Spain, 2016.

1 Introduction

Several models of renewable resource usages by the activities corresponding to different levels of aggregation can be found in the scheduling literature. At the most detailed level, resource units are individualised; an activity requires during all its processing interval a set of resource units (staff, machine, tool) and each resource unit can be assigned at each time to a single activity. From the resource point of view, this allows to define and to track the precise timetable of each individualised resource unit, e.g., by defining for each employee the precise sequence of activities he or she has to achieve within the scheduling horizon. Such resources, most often called machines or disjunctive resources, are widely used in the job shop scheduling literature (Pinedo, 2016). Individualisation of resources is not always necessary nor desirable or even possible. Considering resources that are available in large or even continuously divisible amounts (such as energy or raw material), resource individualisation has no practical sense. Another example: employees that have similar skills and switch from one activity to another under processes. To model these two typical cases, the cumulative resource model defines an aggregated resource as the total number of resource units simultaneously available, without identifying these units. Each activity requires during all its processing a predefined number of resource units, and the cumulative resource constraints check whether, at each time, the total number of units required by the set of activities in process does not exceed the availability. This model is particularly used in the project scheduling literature (Schwindt and Zimmermann, 2015).

In the detailed and aggregated resource models aforementioned, both activities and renewable resources share a common time model. For each activity, the resource units it consumes are required from its start time point to its end time point. However, for some practical applications, a detailed time model is required to define a schedule of activities, whereas the required resources can only be assigned in an aggregated time model. Typical examples can be found in integrated staff rostering and scheduling problem (Artigues et al., 2008; Paul and Knust, 2015) where activities have to be scheduled in a continuous or fine-grain discretised time model and generate a demand in employees that have to be assigned in full shifts (a few hours). On the continuously divisible resource side, energy-efficient scheduling problems (Haït and Artigues, 2011; Gahm et al., 2016) also consider a detailed schedule for the activities (e.g., seconds) while the energy consumption computation is only performed in time intervals of constant length (e.g., a few minutes). Furthermore, project planning at the tactical level may also consider human resource usage on an aggregated basis (e.g., on weeks or months), while project activities are kept on a more precise time reference.

To encompass the above-described applications in a single model, we define a problem for which a set of activities linked by precedence constraints has to be scheduled in a continuous time framework, while resource constraints are checked by averaging the activity consumption on consecutive intervals of identical length. The problem can also be defined as a hybrid project scheduling/capacity planning where the project schedule is established precisely while the capacity is more roughly evaluated on discrete time periods.

After this introduction (Section 1), the paper is structured as follows. Section 2 contains the definition of the problem together with basic structural properties. A comparison with other scheduling problems from the literature is provided in Section 3. A mixed integer linear programming (MILP) formulation is introduced in Section 4. The

adaptation of standard schedule generation schemes (SGS) for the resource-constrained project scheduling problem (RCPSP) is discussed in Section 5. Section 6 proposes to embed both MILP and SGS in an iterative solution scheme (ISS). Experiments and computational results are presented in Section 7. Finally, Section 8 gives some concluding remarks and future work perspectives.

2 PARCPSP – periodically aggregated resource-constrained project scheduling problem

The new problem we are about to define is a variant of the extensively studied RCPSP, in which the demand of activities over aggregated periods is taken into account, to redefine resource constraints.

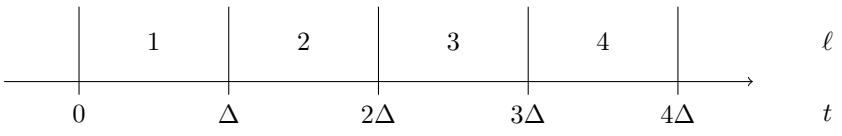
2.1 Definition of the PARCPSP and notations

The input of the problem is a project, composed of activities subject to precedence relations (a predecessor must complete before any of its successors can start). During their execution, activities consume a number of units on one or several resources, each defined by its capacity (limited number of units available). Additionally, a uniform subdivision of the time horizon (i.e., $\mathbb{R}$) into periods of parameterised length $\Delta \in \mathbb{R}_{>0}$ is considered; for all $\ell \in \mathbb{Z}$, the ℓ^{th} period is the time interval $[(\ell-1)\Delta, \ell\Delta]$ (Figure 1). All the notations are given in Table 1.

Table 1 Notations for the input parameters

$\mathcal{A} = \{1, \cdots, n\}$	set of n activities
$\mathcal{R} = \{1, \cdots, m\}$	set of m resources
$p_i \in \mathbb{N}$	processing time of activity $i \in \mathcal{A}$
$b_k \in \mathbb{N}$	capacity of resource $k \in \mathcal{R}$
$r_{i,k} \in \mathbb{N}$	demand/request of activity $i \in \mathcal{A}$ on resource $k \in \mathcal{R}$
$E \subset \mathcal{A} \times \mathcal{A}$	precedence graph
$\Delta \in \mathbb{R}_{>0}$	period length

Figure 1 Uniform subdivision of the time horizon



Resources are supposed renewable: once an activity completes, the resource units it consumed can be reused by other activities immediately. Preemption is not allowed. Two dummy activities are introduced, with a null processing time and null demands on resources. They represent the project beginning (index 0, predecessor of all the activities in $\mathcal{A}$) and the project end (index $n+1$, successor of all the activities in $\mathcal{A}$).

A solution is a vector $S = (S_i)_{1 \leq i \leq n} \in \mathbb{R}^n$, where S_i is the start date of activity $i \in \mathcal{A}$. The start date and the completion date of the project are denoted by $S_0 = \min_{i \in \mathcal{A}}(S_i)$ and $S_{n+1} = \max_{i \in \mathcal{A}}(S_i + p_i)$, respectively.

The objective is to schedule activities so that the project makespan is minimised. Two kinds of constraints are taken into account:

1 For each arc $(i_1, i_2) \in E$, the start date of activity i_2 cannot be lower than the completion date of activity i_1 (precedence constraints).

2 For each resource $k \in \mathcal{R}$, in each period $\ell \in \mathbb{Z}$, the sum of the average requests of the activities cannot exceed the capacity of the resource (periodically aggregated resource constraints).

Let $d_{i,\ell}(S) \in [0, \Delta]$ denote the execution duration of activity $i \in \mathcal{A}$ in period $\ell \in \mathbb{Z}$ depending on solution S, i.e., $d_{i,\ell}(S)$ is the length of the intersection of two intervals: the execution interval of activity i, and period ℓ (Figure 2).

$$\begin{aligned} d_{i,\ell}(S) &= |[S_i, S_i + p_i] \cap [(\ell - 1)\Delta, \ell\Delta]| \\ &= \max\Big(0\ ,\ \min\big(S_i + p_i, \ell\Delta\big) - \max\big(S_i, (\ell - 1)\Delta\big)\Big) \end{aligned}$$

One can note that, given a solution S, the expression of the average request of activity $i \in \mathcal{A}$ on resource $k \in \mathcal{R}$ over period $\ell \in \mathbb{Z}$ is $r_{i,k} \frac{d_{i,\ell}(S)}{\Delta}$.

Therefore, the PARCPSP(Δ) can be formulated as follows:

$$\text{Minimise} \quad S_{n+1} - S_0 \tag{1}$$

$$\text{subject to} \quad S_{i_2} - S_{i_1} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{2}$$

$$\sum_{i \in \mathcal{A}} r_{i,k} \frac{d_{i,\ell}(S)}{\Delta} \leq b_k \qquad \forall k \in \mathcal{R}\ ,\ \forall \ell \in \mathbb{Z} \tag{3}$$

Note that activities may start at any time within a period. In other words, the PARCPSP permits to tackle start and completion events in a precise way, as well as precedence constraints, while the resource consumption is evaluated on average over (aggregated) periods.

Figure 2 Evaluation of the execution duration in aggregated periods

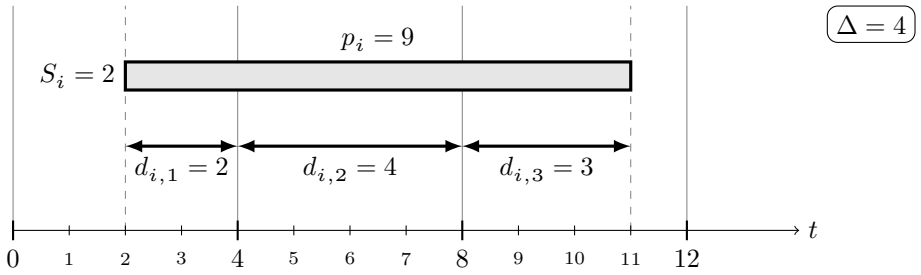
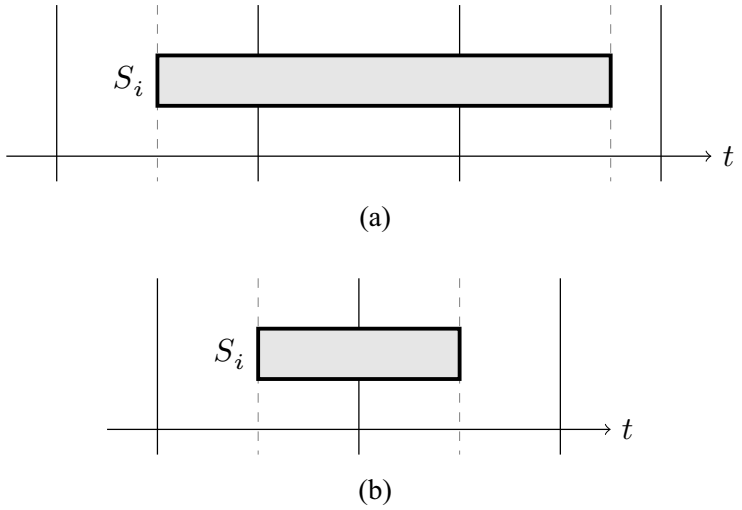


Figure 3 Necessary and sufficient resource feasibility condition depending on the period, (a) first case: $p_i \geq 2\Delta$ (b) second case: $p_i < 2\Delta$ length



We suppose that the project instance respects the following conditions, so that feasible schedules exist, independently of the value of the period length Δ.

- The precedence graph E is acyclic.
- The demand of activity $i \in \mathcal{A}$ on resource $k \in \mathcal{R}$ is not greater than its capacity: $r_{i,k} \leq b_k$

Indeed, for the PARCPSP(Δ), the last condition is sufficient but generally not necessary. Let $S \in \mathbb{R}^n$ a feasible solution. For each activity $i \in \mathcal{A}$, let $\ell_i \in \mathbb{Z}$ the index of the period that contains S_i, i.e., such that $S_i \in [(\ell_i - 1)\Delta, \ell_i\Delta)$.

- If $p_i \geq 2\Delta$, then the period $\ell_i + 1$ is fully included in the execution interval, i.e., $[(\ell_i - 1)\Delta, \ell_i\Delta] \subset [S_i, S_i + p_i]$ [see Figure 3(a)]. So, $d_{i,\ell_i+1}(S) = \Delta$. Since $d_{i,\ell}(S)\frac{r_{i,k}}{\Delta} \leq b_k$ holds in every period ℓ: $r_{i,k} \geq b_k$.
- If $p_i < 2\Delta$, then the execution interval $[S_i, S_i + p_i]$ intersects at most two periods: ℓ_i (always) and $\ell_i + 1$ (possibly). The least restrictive configuration is such that the execution interval is split equally over these two consecutive periods [see Figure 3(b)]. In that case, $d_{i,\ell}(S) = \frac{p_i}{2}$ in both periods ℓ_i and $\ell_i + 1$, zero otherwise. Since $d_{i,\ell}(S)\frac{r_{i,k}}{\Delta} \leq b_k$ holds in every period ℓ: $r_{i,k} \geq b_k \frac{\Delta}{p_i/2}$.

As a result, a necessary and sufficient resource feasible condition is:

$$r_{i,k} \leq b_k \cdot \max\left(1\ ,\ \tfrac{2\Delta}{p_i}\right)$$

Note that, apart from the period length Δ, the input of the RCPSP is the same as the input of the PARCPSP.

Figure 4 Evaluation of activity demands on resources (see online version for colours)

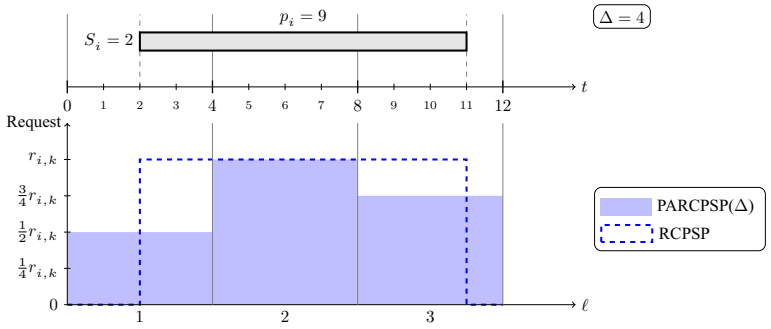
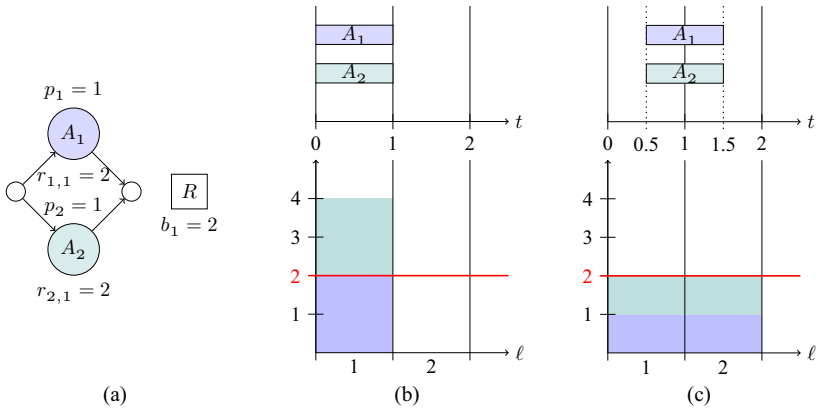


Figure 5 Impact of translation on the feasibility of a schedule, (a) instance (b) infeasible solution (c) feasible solution (see online version for colours)



Let $\mathcal{A}_t(S) \subseteq \mathcal{A}$ denote the set of activities in progress at $t \in \mathbb{R}$ depending on solution S.

$$\mathcal{A}_t(S) = \left\{ i \in \mathcal{A} \;\middle|\; t \in [S_i, S_i + p_i) \right\}$$

A possible formulation for the RCPSP is:

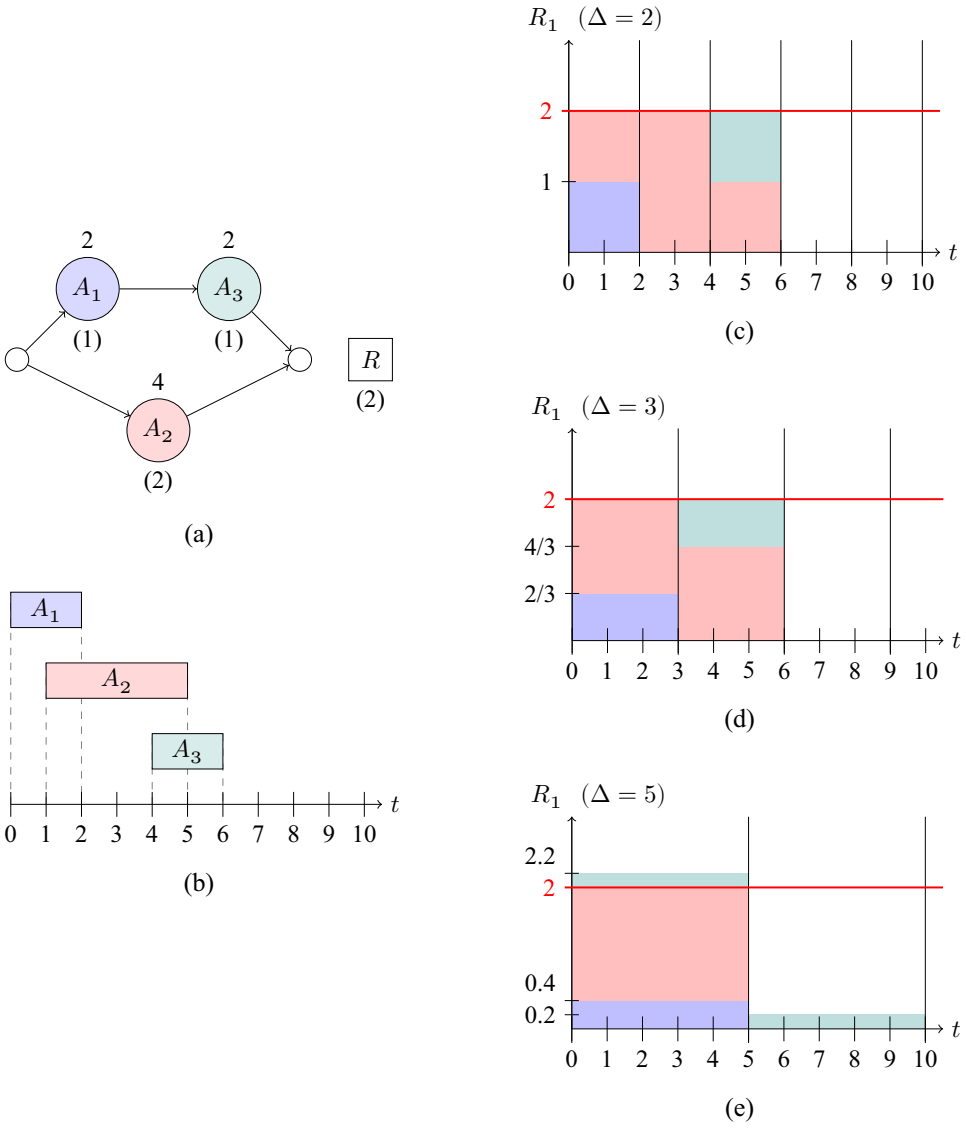
$$\text{Minimise} \quad S_{n+1} - S_0 \tag{4}$$

$$\text{subject to} \quad S_{i_2} - S_{i_1} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{5}$$

$$\sum_{i \in \mathcal{A}_t(S)} r_{i,k} \leq b_k \qquad \forall k \in \mathcal{R}\ ,\ \forall t \in \mathbb{R} \tag{6}$$

The RCPSP consists in minimising the project makespan (1) under precedence constraints (2) and resource constraints (3) (at each instant). The only difference between the PARCPSP(Δ) and the RCPSP lies in the definition of the resource constraints, that are handled on average or instantaneously, respectively (Figure 4).

Figure 6 Impact of the period length on the feasibility of a schedule, (a) instance (b) solution (c) feasible (optimal) solution when $\Delta = 2$ (d) feasible (non-optimal) solution when $\Delta = 3$ (e) infeasible solution when $\Delta = 5$ (see online version for colours)



2.2 Properties

Proposition 1: For all $\Delta \in \mathbb{R}_{>0}$, the PARCPSP(Δ) is a relaxation of the RCPSP.

Proof: Let $\Delta \in \mathbb{R}_{>0}$. Let $S \in \mathbb{R}^n$ a feasible solution for the RCPSP: S satisfies both precedence constraints and resource constraints (at each instant).
For all $i \in \mathcal{A}$, for all $t \in \mathbb{R}$, let $\alpha_{i,t}(S) = 1$ if $i \in \mathcal{A}_t(S)$, 0 otherwise. Since $\mathcal{A}_t(S) = \left\{ i \in \mathcal{A} \;\middle|\; \alpha_{i,t}(S) = 1 \right\}$, the resource constraints from the RCPSP can be rewritten as:

$$\sum_{i \in \mathcal{A}} r_{i,k} \alpha_{i,t}(S) \leq b_k \qquad \forall k \in \mathcal{R} \ , \ \forall t \in \mathbb{R}$$

Moreover, $d_{i,\ell}(S)$ can be expressed directly from $\alpha_{i,t}(S)$:

$$d_{i,\ell}(S) = \int_{(\ell-1)\Delta}^{\ell\Delta} \alpha_{i,t}(S)\, dt \qquad \forall i \in \mathcal{A} \ , \ \forall \ell \in \mathbb{Z}$$

Therefore:

$$\sum_{i \in \mathcal{A}} r_{i,k} d_{i,\ell}(S) = \int_{(\ell-1)\Delta}^{\ell\Delta} \left(\sum_{i \in \mathcal{A}} r_{i,k} \alpha_{i,t}(S) \right) dt \leq b_k \Delta \qquad \forall k \in \mathcal{R} \ , \ \forall \ell \in \mathbb{Z}$$

Since S also satisfies the aggregated resource constraints (in each period), it is also a feasible solution for the PARCPSP(Δ). ■

Before continuing, let us recall some properties that characterise solutions of the standard RCPSP [for more details, the reader is kindly referred to Artigues et al. (2008) and Schwindt and Zimmermann (2015)].

- Translating a solution $S \in \mathbb{R}^n$, i.e., defining a new solution $S' \in \mathbb{R}^n$ such that the difference $S'_i - S_i$ is the same for all $i \in \{1, \cdots, n\}$, has no impact on its feasibility.
- Non-negative integer solutions ($\forall i \in \mathcal{A}$, $S_i \in \mathbb{N}$) are dominant, i.e., it always exists an optimal integer solution.

In the case of the PARCPSP, because the resource profile is established for each aggregated period from the average requests of activities, such solution classes are no more dominant. Moreover, depending on the project data, the set of feasible schedules may be enlarged (strictly), even with aggregated period of unit length ($\Delta = 1$). These observations are formalised in the next properties.

Proposition 2: In the case of the PARCPSP(Δ) (for all $\Delta \in \mathbb{R}_{>0}$), translating a solution may have an impact on its feasibility.

Proposition 3: In the case of the PARCPSP(Δ) (for all $\Delta \in \mathbb{R}_{>0}$), integer solutions are not dominant.

Proposition 4: The PARCPSP(1) is still a relaxation of the RCPSP (even with unit periods, i.e., $\Delta = 1$).

Proof: An example that illustrates Propositions 2 to 4 is given in Figure 5.
In the case of the RCPSP, it is not possible to process both activities in parallel. When there is no idle time between the two activities, the minimum makespan (equal to 2) is reached.

In the case of the PARCPSP(1), a solution such that both activities are processed in parallel may or may not be feasible, depending on how the execution intervals intersect the aggregated periods. Hence, the minimum makespan is 1 (reachable only with non-integer start dates). ■

Proposition 5: In the general case, reducing the value of the parameter Δ does not define a relaxation closer to the RCPSP. Formally, given $\Delta_1 \in \mathbb{R}_{>0}$ and $\Delta_2 \in \mathbb{R}_{>0}$ such that $\Delta_1 < \Delta_2$, the PARCPSP(Δ_2) is generally not a relaxation of the PARCPSP(Δ_1).

Proof: A counter-example is presented in Figure 6. The feasibility of the schedule $S = (0, 1, 4)$ depends on the period length Δ :

- S is feasible for the PARCPSP(2) (indeed, it is even optimal).
- S is feasible for the PARCPSP(3) (but it is not optimal; the schedule $S' = (1, 1, 3)$, for which the average request is the same as S in all periods, is optimal).
- S is not feasible for the PARCPSP(5). ■

3 Literature review

Among project scheduling models at the tactical level, well-suited for the management of aggregated data, an additional concept is often taken into account, the intensity of an activity, that allows a flexible resource usage. On the one hand, in each period, the lower the intensity, the less resource units are required. On the other hand, the completion of an activity is determined by the sum of the intensities. Therefore, for each activity, the number of periods intersected also has to be decided.

For instance, in the rough cut capacity planning (RCCP), introduced by Hans (2001), activity start and completion events can occur at any time (not only at period bounds). The methodology implemented consists of a branch-and-price, with the generation of feasible project plans indicating in which periods activities may be processed, while the determination of the intensities in each period in order to respect the capacity constraints is managed in the master problem. However, only activity intensities per period are considered, while two precedence-related activities may be executed in the same period; the model does not prevent overlaps, thus potentially leading to infeasible schedules with respect to precedence constraints. Gademann and Schutten (2005) have proposed linear programming based heuristics to solve this problem. Mestry et al. (2011) use a similar approach as Hans' for job-shop production in a make-to-order context.

One possibility to ensure that precedence constraints are always satisfied, is to over-constrain the RCCP, by enforcing that, if an activity i completes its execution in period ℓ, then all its successors may start only from the next period $\ell + 1$ on. This leads to the definition of another problem, first introduced by Kis (2005), entitled the resource-constrained project scheduling problem with variable-intensity activities

(RCPSVP). Kis (2005) proposes several MILP formulations together with a polyhedral study for the the generation of valid inequalities deduced from predecessor-successor relations incorporated into a branch-and-cut approach.

Nonetheless, it is possible to overcome the precedence issue without over-constraining the RCCP, i.e., allowing that a predecessor and a successor respectively complete and start in the same period, with no overlap. This has been achieved by Haït and Baydoun (2012) who proposed a MILP formulation based on a mixed time representation, introducing for each activity continuous variables representing the start and completion dates, independently of the subdivision of the planning horizon into periods. The main difficulty lies in linking the continuous time variables with discrete time variables linked to periods, which is solved by reinterpreting some of the temporal relations defined in Allen's interval algebra, an abstraction of all the possible relative positioning of two time windows (Allen, 1981).

In the chemical engineering field, researchers have investigated the respective merits of continuous-time and discrete-time models for process scheduling (Floudas and Lin, 2004). However, the literature is not rich in mixed-time models. Energy problems provide a good framework to develop such models: energy cost varies on a discrete way while the activities may start or finish at any time. Castro et al. (2009, 2011) propose a formulation where continuous-time events are located within fixed time intervals defined by global time points. Finally, Silvente et al. (2015) propose a mixed time model for task scheduling within an energy grid.

The problem considered in this paper, namely the PARCPSP, can be seen as a special case of the RCCP, where the intensity of an activity is fixed, i.e., its resource usage in a period is determined by the execution time in this period. In particular, this implies that no preemption is possible, since the intensity in an intersected period cannot be null. The MILP formulation described in the next section is inspired from the one of Haït and Baydoun (2012). Here, since the processing times are deterministic, additional relations can be deduced to link the binary variables that identify the period in which an activity starts/completes.

4 Mixed integer linear problem formulation based on a mixed-time representation

4.1 Decision variables: a mixed-time representation

The implementation requires a finite number of periods $L \in \mathbb{N}$. Let us consider the set of consecutive periods $\mathcal{L} = \{1, \cdots, L\}$. In a similar fashion, standard time-indexed models for the RCPSP [see e.g., Schwindt and Zimmermann (2015), Artigues et al. (2008)] make use of an extra parameter $H \in \mathbb{N}$, the time horizon, i.e., an upper bound on the shortest project duration (the objective function), that represents the number of unit periods considered. In the case of the PARCPSP(Δ), L can be set to $\left\lceil \frac{H}{\Delta} \right\rceil$.

The decision variables are listed in Table 2. Two temporal representations coexist: a continuous time representation (S_i variables) and a discrete time representation ($zs_{i,\ell}$ and $zf_{i,\ell}$ variables) as shown in Figure 7. These two representations are needed to compute the value of $d_{i,\ell}$ variables, required in aggregated resource constraints.

Table 2 Description of the decision variables

$S_i \in [0, L\Delta]$	start date of activity $i \in \mathcal{A} \cup \{0, n+1\}$
$zs_{i,\ell} \in \{0,1\}$	step binary variable equal to 1 *iff* activity $i \in \mathcal{A}$ starts in period $\ell \in \mathcal{L}$ or before
$zf_{i,\ell} \in \{0,1\}$	step binary variable equal to 1 *iff* activity $i \in \mathcal{A}$ finishes in period $\ell \in \mathcal{L}$ or before
$d_{i,\ell} \in [0, \Delta]$	execution duration of activity $i \in \mathcal{A}$ in period $\ell \in \mathcal{L}$ (introduced previously)

Figure 7 Link between all decision variables (mixed time representation)

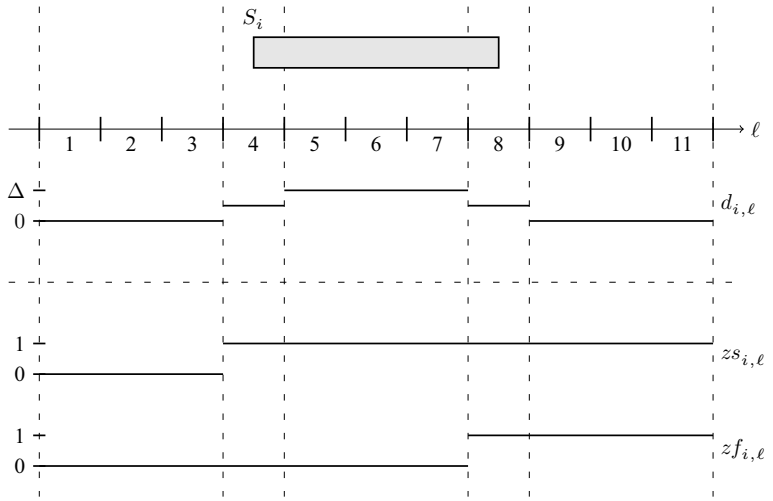
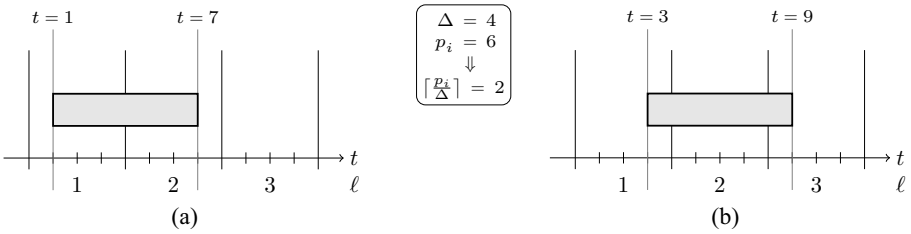


Figure 8 Two possible configurations on the number of periods intersected (definition of π_i), (a) number of periods intersected minimal ($\pi_i = 0$) (b) number of periods intersected maximal ($\pi_i = 1$)



(a) (b)

4.2 Objective and constraints (initial formulation)

The MILP formulation for the PARCPSP(Δ) is given hereafter.

$$\text{Minimise} \quad S_{n+1} - S_0 \tag{1}$$

$$\text{subject to} \quad S_{i_2} - S_{i_1} \geq p_{i_1} \qquad \forall (i_1, i_2) \in E \tag{2}$$

$$\sum_{i=1}^{n} r_{i,k} d_{i,\ell} \leq b_k \Delta \qquad \forall k \in \mathcal{R}\ ,\ \forall \ell \in \mathcal{L} \tag{3}$$

$$S_i \geq \ell\Delta(1 - zs_{i,\ell}) \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{4}$$

$$S_i \leq L\Delta - (L - \ell)\Delta zs_{i,\ell} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{5}$$

$$S_i + p_i \geq \ell\Delta(1 - zf_{i,\ell}) \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{6}$$

$$S_i + p_i \leq L\Delta - (L - \ell)\Delta zf_{i,\ell} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{7}$$

$$d_{i,\ell} \geq 0 \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{8}$$

$$d_{i,\ell} \leq \Delta(zs_{i,\ell} - zf_{i,\ell-1}) \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{9}$$

$$d_{i,\ell} \geq \Delta(zs_{i,\ell-1} - zf_{i,\ell}) \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{10}$$

$$d_{i,\ell} \geq S_i + p_i - (\ell - 1)\Delta - \Delta(1 - zs_{i,\ell-1}) - (L - \ell + 1)\Delta(1 - zf_{i,\ell}) \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{11}$$

$$d_{i,\ell} \geq \ell\Delta - S_i - \Delta zf_{i,\ell} - \ell\Delta zs_{i,\ell-1} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{12}$$

$$\sum_{\ell=1}^{L} d_{i,\ell} = p_i \qquad \forall i \in \mathcal{A} \tag{13}$$

$$zs_{i,\ell-1} \leq zs_{i,\ell} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{14}$$

$$zf_{i,\ell-1} \leq zf_{i,\ell} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{15}$$

$$zs_{i,\ell} \in \{0, 1\} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{16}$$

$$zf_{i,\ell} \in \{0, 1\} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{17}$$

The objective (1) and the two first constraints (2) and (3) match the definition of the PARCPSP: minimise the project duration under precedence constraints and periodically aggregated resource constraints.

Constraints (4) and (5) [respectively (6) and (7)] ensure the coherence of the two temporal representations: $zs_{i,\ell}$ (respectively $zf_{i,\ell}$) step occurs in the period that contains the start date S_i (respectively the completion date $S_i + p_i$) for each activity $i \in \mathcal{A}$.

The next constraints enable the computation of $d_{i,\ell}$ values. Constraints (8) and (9) state that $d_{i,\ell} \in [0, \Delta]$. Constraints (9) set $d_{i,\ell}$ to 0 in periods ℓ that either precede or follow the execution interval (i.e., $(S_i, S_i + p_i) \cap ((\ell - 1)\Delta, \ell\Delta) = \emptyset$). Constraints (10) set $d_{i,\ell}$ to Δ in periods ℓ that are fully included in the execution interval (i.e., $[(\ell - 1)\Delta, \ell\Delta] \subseteq [S_i, S_i + p_i]$). Constraints (11) provide a lower bound on the value of $d_{i,\ell}$ in the period ℓ that contains the completion date (i.e., $S_i + p_i \in [(\ell - 1)\Delta, \ell\Delta]$). Constraints (12) provide a lower bound on the value of $d_{i,\ell}$ in the period ℓ that contains

the start date (i.e., $S_i \in [(\ell-1)\Delta, \ell\Delta]$). Constraints (13) ensure that activity $i \in \mathcal{A}$ is processed during exactly p_i time units.

Constraints (14) and (15) enforce an increasing step behaviour on binary variables; their integrity is stated in constraints (16) and (17).

Remark: Both expressions $zs_{i,\ell}$ and $zf_{i,\ell}$ have to be replaced with 0 if $\ell < 1$ and with 1 if $\ell > L$, since activities cannot start before $t = 0$ (beginning of the first period 1) nor complete after $t = L\Delta$ (end of the last period L). In other words, binary variables linked to out-of-scope periods do not need to be created; however, their value can be used in the model.

4.3 Improvements of the formulation

4.3.1 Lower and upper bounds on start dates

For each activity $i \in \mathcal{A}$, it is possible to compute an earliest start date ES_i and a latest start date LS_i during a preprocessing phase [typically obtained by computing longest paths in the precedence graph, or by propagation algorithms, see e.g., Artigues et al. (2008)], that can be then incorporated directly in the model.

$$ES_i \leq S_i \leq LS_i \qquad \forall i \in \mathcal{A} \tag{18}$$

Moreover, the fact that the length of all the periods is equal to Δ induce symmetries: the feasibility of a solution is not affected by a translation of $x\Delta$ for all $x \in \mathbb{Z}$. This phenomenon can be avoided by enforcing the project to start in the first period. This can be achieved with a single constraint:

$$S_0 \leq \Delta \tag{19}$$

4.3.2 Taking into account the number of periods intersected

Indeed, constraints (17) (integrity of $zf_{i,\ell}$ variables) can be replaced with the following constraints.

$$0 \leq zf_{i,\ell} \leq 1 \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{17'}$$

$$zs_{i,\ell} \geq zf_{i,\ell+\lceil \frac{p_i}{\Delta} \rceil - 1} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{20}$$

$$zs_{i,\ell} \leq zf_{i,\ell+\lceil \frac{p_i}{\Delta} \rceil} \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{21}$$

$$\pi_i \in \{0,1\} \qquad \forall i \in \mathcal{A} \tag{22}$$

$$zs_{i,\ell} \leq zf_{i,\ell+\lceil \frac{p_i}{\Delta} \rceil - 1} + \pi_i \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{23}$$

$$zs_{i,\ell} \geq zf_{i,\ell+\lceil \frac{p_i}{\Delta} \rceil} + \pi_i - 1 \qquad \forall i \in \mathcal{A}\ ,\ \forall \ell \in \mathcal{L} \tag{24}$$

Note that $zf_{i,\ell}$ variables are now implemented as continuous variables, after constraints (17'). This permits to considerably reduce the number of binary variables subject to an explicit integrity constraint (about twice as less with a large number of

periods L). The additional constraints ensure that $zf_{i,\ell}$ variables always take an integer value.

Since the subdivision of the time horizon is uniform, and the processing times are deterministic, only two possible configurations exist for the execution interval of any activity $i \in \mathcal{A}$ in any solution S.

1 $[S_i, S_i + p_i]$ intersects at least $\left\lceil \frac{p_i}{\Delta} \right\rceil$ periods [constraints (20)].

2 $[S_i, S_i + p_i]$ intersects at most $\left\lceil \frac{p_i}{\Delta} \right\rceil + 1$ periods [constraints (21)].

Thus, it is possible to introduce, for each activity $i \in \mathcal{A}$, one binary variable π_i [constraints (22)] that describes which configuration among the two aforementioned ones applies (see also Figure 8).

1 $\pi_i = 0$ *iff* $[S_i, S_i + p_i]$ intersects exactly $\left\lceil \frac{p_i}{\Delta} \right\rceil$ periods [constraints (23)].

2 $\pi_i = 1$ *iff* $[S_i, S_i + p_i]$ intersects exactly $\left\lceil \frac{p_i}{\Delta} \right\rceil + 1$ periods [constraints (24)].

5 Adaptation of SGS

For the RCPSP, standard algorithms enable to generate a solution from a priority list, i.e., a permutation σ of activities, such that '$\sigma(j) = i$' means 'activity $i \in \mathcal{A}$ is at position $j \in \{1, \cdots, n\}$', that is a linear extension of the partial order defined by E (precedence relations):

$$\forall (i_1, i_2) \in E \qquad \sigma^{-1}(i_1) < \sigma^{-1}(i_2)$$

We propose an adaptation of two standard SGS to the PARCPSP: the serial schedule generation scheme (SSGS), that consists in scheduling successively each activity as early as possible, and the parallel schedule generation scheme (PSGS), that consists in filling successively each unit period as much as possible [for a detailed presentation, see e.g., Artigues et al. (2008)].

In the case of the RCPSP, optimal implementations are based on start/completion events. However, in the case of the PARCPSP, the temporal approach is completely modified: an evaluation of the average resource usage per aggregated period is required. This is why some concepts have to be discussed, before entering the details of the adapted SGS.

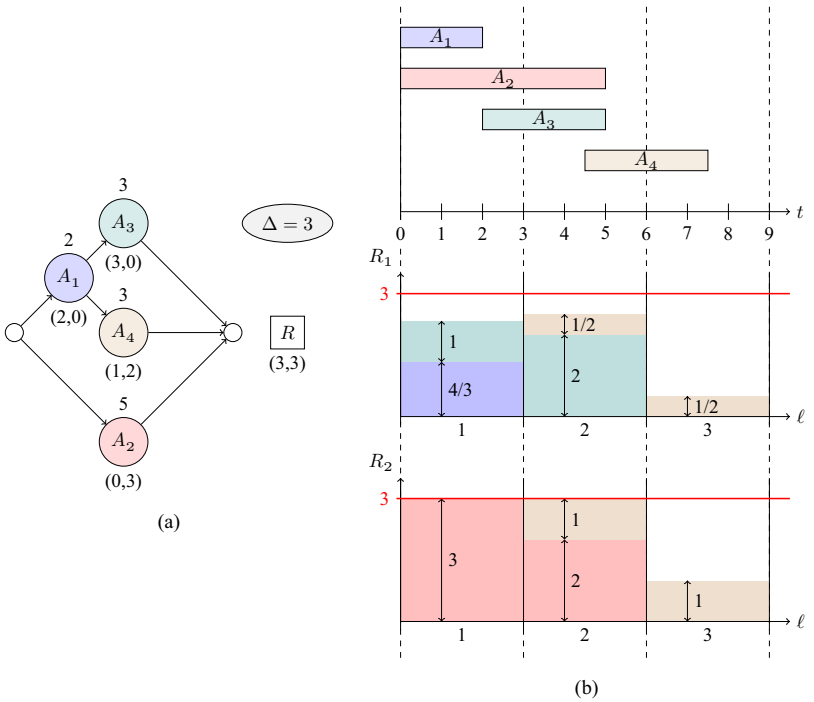
5.1 Underlying concepts

5.1.1 Resource feasibility test for an activity at a given date

Let us suppose activities in set $\mathcal{A}' \subset \mathcal{A}$ have already been scheduled, i.e., for all $i' \in \mathcal{A}'$, start dates $S_{i'}$ (and so execution durations $d_{i',\ell}(S)$, $\ell \in \mathbb{Z}$) have been fixed previously.

Let $i \in \mathcal{A} \backslash \mathcal{A}'$ the next activity to schedule ($\mathcal{A}'$ is supposed to contain all the predecessors of i).

Figure 9 Example – serial schedule generation scheme, (a) instance (b) solution generated by the SSGS (see online version for colours)



Given a candidate start date $t \in \mathbb{R}$ for activity i (not lower than the latest completion time among predecessors of i, i.e., t respects precedence constraints), can i effectively start at t (i.e., are resource constraints satisfied)?

Clearly, the answer depends on the resource availability in the periods intersected by the candidate execution interval $[t, t + p_i]$.

Let $\overline{d_{i,\ell}}(\mathcal{A}')$ denote the maximum value of $d_{i,\ell}(S)$ for activity i to be scheduled in period $\ell \in \mathbb{Z}$, depending on activities already scheduled in $\mathcal{A}'$. Let $\mathcal{L}_i(t)$ denote the set of consecutive periods intersected by the candidate execution interval, i.e., $\mathcal{L}_i(t) = \left\{\ell \in \mathbb{Z} \;\middle|\; [(\ell-1)\Delta, \ell\Delta] \cap [t, t+p_i] \neq \emptyset\right\}$.

$$\text{Activity } i \text{ can effectively start at } t \quad \Leftrightarrow \quad \forall \ell \in \mathcal{L}_i(t) \quad d_{i,\ell}(S) \leq \overline{d_{i,\ell}}(\mathcal{A}')$$

5.1.2 Computation of the maximum execution duration

According to the aggregated resource constraints, and considering only activities in $\mathcal{A}' \cup \{i\}$:

$$\sum_{i' \in \mathcal{A}'} r_{i',k} d_{i',\ell}(S) + r_{i,k} d_{i,\ell}(S) \leq b_k \Delta \qquad \forall k \in \mathcal{R}$$

Therefore:

$$\overline{d_{i,\ell}}(\mathcal{A}') = \min\left(\Delta\ ,\ \min_{k\in\mathcal{R}\ |\ r_{i,k}>0}\left(\frac{b_k\Delta - \sum_{i'\in\mathcal{A}'} r_{i',k}d_{i',\ell}(S)}{r_{i,k}}\right)\right)$$

5.2 SGS algorithms

The pseudo-codes of the adapted SSGS and the adapted PSGS are given in Algorithm 1 and Algorithm 2, respectively.

Algorithm 1 Adaptation of the SSGS

/* Refer to the underlying concepts for the test done at line 13 and the notation $\overline{d_{i,\ell}}(\mathscr{A}')$. */
/* $\Gamma_i^{\ominus}$ is the set of the predecessors of activity $i \in \mathscr{A} : \Gamma_i^{\ominus} = \left\{i^{\ominus} \in \mathscr{A} \mid (i^{\ominus}, i) \in E\right\}$ */

1 $\mathscr{A}' \leftarrow \emptyset$ /* Set of activities scheduled */
2 **for** $j = 1$ **to** n **do**
3 $\quad i \leftarrow \sigma(j)$ /* Activity i is at position j */
4 $\quad t_{\min} \leftarrow \max_{i^{\ominus}\in\Gamma_i^{\ominus}}(S_{i^{\ominus}} + p_{i^{\ominus}})$ /* Greatest completion date among predecessors */
5 $\quad \ell \leftarrow \max(1, 1 + \lfloor \frac{t_{\min}}{\Delta} \rfloor)$
6 $\quad S_i \leftarrow \max(t_{\min}, \ell\Delta - \overline{d_{i,\ell}}(\mathscr{A}'))$
7 $\quad$ **if** $S_i = \ell\Delta$ **then** /* $\overline{d_{i,\ell}}(\mathscr{A}') = 0$: activity i cannot start in period ℓ */
8 $\quad\quad$ **repeat**
9 $\quad\quad\quad \ell \leftarrow \ell + 1$
10 $\quad\quad\quad S_i \leftarrow \ell\Delta - \overline{d_{i,\ell}}(\mathscr{A}')$
11 $\quad\quad$ **until** $S_i < \ell\Delta$ /* $\overline{d_{i,\ell}}(\mathscr{A}') > 0$: activity i can possibly start in period ℓ */
12 $\quad$ **end if**
13 $\quad$ **while** *Activity i cannot effectively start at S_i* **do**
14 $\quad\quad$ **repeat**
15 $\quad\quad\quad \ell \leftarrow \ell + 1$
16 $\quad\quad\quad S_i \leftarrow \ell\Delta - \overline{d_{i,\ell}}(\mathscr{A}')$
17 $\quad\quad$ **until** $S_i < \ell\Delta$ /* $\overline{d_{i,\ell}}(\mathscr{A}') > 0$: activity i can possibly start in period ℓ */
18 $\quad$ **end while**
19 $\quad$ **if** $p_i \leqslant \ell\Delta - S_i$ **then** /* Only one aggregated period intersected */
20 $\quad\quad S_i \leftarrow \max(t_{\min}, (\ell-1)\Delta)$ /* Left shift */
21 $\quad$ **end if**
22 $\quad \mathscr{A}' \leftarrow \mathscr{A}' \cup \{i\}$ /* Update of the set of activities scheduled */
23 **end for**

Algorithm 2 Adaptation of the PSGS

/* Refer to the underlying concepts for the test done at line 11 and the notation $\overline{d_{i,\ell}}(\mathscr{A}')$. */
/* $\Gamma_i^{\ominus}$ is the set of the predecessors of activity $i \in \mathscr{A} : \Gamma_i^{\ominus} = \{i^{\ominus} \in \mathscr{A} \mid (i^{\ominus}, i) \in E\}$ */

1 $\mathscr{A}' \leftarrow \emptyset$ /* Set of activities scheduled */
2 $\ell \leftarrow 1$ /* Index of the first aggregated period considered */
3 **while** $\mathscr{A}' \neq \mathscr{A}$ **do**
4 $\quad i^* \leftarrow 0$ /* Convention; means "no activity found" */
5 $\quad t^* \leftarrow \ell\Delta$ /* Candidate start date for activity i^* */
6 $\quad$ **for** $j = 1$ **to** n **do**
7 $\quad\quad i \leftarrow \sigma(j)$ /* Activity i is at position j */
8 $\quad\quad$ **if** $i \notin \mathscr{A}'$ **and** $\Gamma_i^{\ominus} \subseteq \mathscr{A}'$ **then**
$\quad\quad\quad$ /* Activity i has not already been scheduled */
$\quad\quad\quad$ /* All the predecessors of activity i have been scheduled */
9 $\quad\quad\quad t_{\min} \leftarrow \max_{i^{\ominus} \in \Gamma_i^{\ominus}}(S_{i^{\ominus}} + p_{i^{\ominus}})$ /* Greatest completion date among predecessors */
10 $\quad\quad\quad t \leftarrow \max(t_{\min}, \ell\Delta - \overline{d_{i,\ell}}(\mathscr{A}'))$
11 $\quad\quad\quad$ **if** $t < t^*$ **and** *Activity i can effectively start at t* **then**
12 $\quad\quad\quad\quad i^* \leftarrow i$
13 $\quad\quad\quad\quad t^* \leftarrow t$
14 $\quad\quad\quad\quad t^*_{\min} \leftarrow t_{\min}$
15 $\quad\quad\quad$ **end if**
16 $\quad\quad$ **end if**
17 $\quad$ **end for**
18 $\quad$ **if** $i^* = 0$ **then** /* No activity found */
19 $\quad\quad \ell \leftarrow \ell + 1$ /* Move to the next aggregated period */
20 $\quad$ **else**
21 $\quad\quad$ **if** $p_i \leqslant \ell\Delta - t^*$ **then** /* Only one aggregated period intersected */
22 $\quad\quad\quad S_{i^*} \leftarrow \max(t^*_{\min}, (\ell-1)\Delta)$ /* Left shift */
23 $\quad\quad$ **else**
24 $\quad\quad\quad S_{i^*} \leftarrow t^*$
25 $\quad\quad$ **end if**
26 $\quad\quad \mathscr{A}' \leftarrow \mathscr{A}' \cup \{i^*\}$ /* Update of the set of activities scheduled */
27 $\quad$ **end if**
28 **end while**

5.3 Example

An example application of the SSGS is proposed in Figure 9. The project instance is composed of $n = 4$ activities and $m = 2$ resources. The priority list σ considered is the identity.

1 The first activity has a positive demand only on the first resource. Starting from $t = 0$, its execution interval intersects only the first aggregated period; by definition, its average request is equal to $r_{1,1}\frac{d_{1,1}}{\Delta} = 2 \times \frac{2}{3} = \frac{4}{3}$.

2 The second activity has a positive demand only on the second resource; so, it can also start at $t = 0$. This time, two aggregated periods are intersected.

3 The third activity must start after the first activity. The precedence constraint is dominant here: the activity can start at $t = 2$ (note that, in the first aggregated period, the sum of the average requests on the first resource remains strictly lower than the capacity).

4 The fourth activity must also start after the first activity. However, this time, the aggregated resource constraint on the second resource prevents the activity from starting right after its predecessor; indeed, it cannot start before the middle of the second aggregated period, at $t = 4.5$.

6 Iterative solution scheme

Generally speaking, the different methodologies from the literature can be classified into two categories: on the one hand, exact methods, for instance based on MILP formulations, and on the other hand heuristic, generally building an initial schedule and then transforming it until no better schedule is generated or after a fixed number of iterations. Here, we propose an original approach, at the frontier of these two families.

The number of aggregated periods considered in the implementation is $L = \lceil \frac{H}{\Delta} \rceil$. Therefore, the size of the MILP (both the number of binary variables and constraints) is inversely proportional to the period length Δ: the smaller Δ, the more computational time needed to solve to optimality.

In order to speed up the resolution process for a PARCPSP problem defined with a small period length, the MILP and the SGSs can be embedded in an ISS. The main idea is to solve successively several PARCPSP problems, while decreasing progressively the period length.

Let $\Delta_F \in \mathbb{R}_+^*$ the period length of the final problem to solve, namely, the PARCPSP(Δ_F). An ISS can be set up in the following way. Let $\Delta_0, \Delta_1, \cdots, \Delta_F$ a list of period lengths, sorted in descending order.

1 Solve the PARCPSP(Δ_0) using the MILP.

2 For each $\Delta \in \{\Delta_1, \cdots, \Delta_F\}$ in descending order:

   a Generate a pool of feasible solutions for the PARCPSP(Δ), by repairing solutions obtained from the previous resolutions of problems PARCPSP(Δ') with $\Delta' > \Delta$.
   The repair process is composed of two steps.

      1 Transform the solution to repair S into a priority list σ, such that:

$$\forall (j_1, j_2) \in \{1, \cdots, n\}^2 \qquad j_1 \leq j_2 \quad \Rightarrow \quad S_{\sigma(j_1)} \leq S_{\sigma(j_2)}$$

      2 Apply an SGS onto σ that returns the repaired solution S'.

   More solutions can be added in the pool by considering more priority lists, obtained by applying a `SWAP` operator restricted to consecutive activities. More precisely, given a position $j \in \{1, \cdots, n-1\}$ and a priority list σ, let $i_1 \leftarrow \sigma(j)$ and $i_2 \leftarrow \sigma(j+1)$; unless $(i_1, i_2) \in E$, the `SWAP` operator

exchanges the two activities ($\sigma(j) \leftarrow i_2$ and $\sigma(j+1) \leftarrow i_1$) with a probability ρ.

b Solve the PARCPSP(Δ) using the MILP, providing generated solutions from the pool as hints (upper bounds) to the solver (warm start).
Contrarily to what is stated in Morin et al. (2016), it is indeed not possible to maintain a non-decreasing lower bound throughout the process (a counterexample is provided in Proposition 5).

The efficiency of this approach is determined by two parameters: the number of iterations and the reduction of the period length between successive iterations. The smaller Δ, the less reduction should be applied, in order to propagate pertinent information.

7 Experiments and results

7.1 Experiments description

Three different methods have been tested to solve a problem PARCPSP(Δ).

1 The first method `[M1]` consists in applying both SSGS and PSGS on a large number of priority lists.

2 The second method `[M2]` consists in solving the MILP; the solver is given the solutions obtained from `[M1]` as hints (MIP warm start).

3 The third method `[M3]` consists in setting up an ISS as described in Section 6. Several PARCPSP problems are considered; from one iteration to another, the period length decreases, and the initial solution pool is updated through a repair process.

The methods have been tested in the following conditions:

- The J30 instances from the PSPLIB (Kolisch and Sprecher, 1996) have been used: 480 projects with 30 activities and 4 resources each.
- The maximal number of priority lists considered has been set to 1,000; the initial priority list is the identity (activities are numbered in such a way that this is always a valid priority list) that is then modified using the `SWAP` operator (parameter ρ set to 20%).
- The solver used for the resolution of the MILP is CPLEX 12.6.2 (IBM).
- A timeout of 3,600 seconds has been set for each MILP resolution.

7.2 Results analysis

A first series of experimental results aiming at comparing the three methods are available in Table 3. For all the methods, the gap to the best upper bound found, as well as the computational time, are reported. For both `[M1]` and `[M2]`, the number of optimal solutions is also indicated.

It is important to mention that, for `[M3]`, the results are *cumulative*, i.e., for the r-th row:

- The problem solved is the PARCPSP(Δ_r) (the value of Δ_r can be read in the leftmost column).
- The ISS is built with r iterations: the values of period lengths are $\Delta_1, \Delta_2, \cdots, \Delta_r$ (previous rows).
- The results in Table 3 take into account the execution of *all* the iterations.

Table 3 Experimental results

Δ	*[M1]*		*[M2]*			*[M3]*		
	Gap UB (%)	*Time (s)*	*Gap UB (%)*	*Time (s)*	*# Opt (/480)*	*Gap UB (%)*	*Time (s)*	*# Opt (/480)*
10	17.23	< 1	0	75	480	0	75	480
8	14.16	< 1	0	89	480	0	117	480
6	12.01	< 1	0.46	138	475	0	151	475
5	9.69	< 1	1.16	154	458	0	169	460
4	8.55	< 1	1.83	207	450	0	186	452
4.5	7.49	< 1	2.17	223	443	0	195	447
3	6.98	< 1	2.49	231	442	0	208	444
2.75	6.3	< 1	2.92	241	437	0	223	441
2.5	6.07	< 1	3.27	250	435	0	241	439
2.25	5.72	< 1	4.12	277	433	0	283	436
2	5.29	< 1	3.88	296	431	0	320	436
1.8	4.96	< 1	4.35	339	428	0	381	435
1.6	4.61	< 1	3.28	372	427	0	447	435
1.4	4.29	< 1	3.55	421	426	0	532	434
1.3	3.84	< 1	3.41	463	423	0	599	434
1.2	3.5	< 1	2.99	488	421	0	657	433
1.1	3.27	< 1	3.11	517	418	0	724	432
1	3.07	< 1	3.03	535	417	0	795	429

In terms of gap to the best upper bound, the best solutions are always found by `[M3]`. The gap of `[M1]` is quite large, especially for large values of Δ. This can be explained by the fact that the solutions built by both SGS systematically start at $t = 0$; nonetheless, there is no guarantee at least one such solution be optimal (cf., proof of Propositions 2 to 4 for example). In terms of computational time, the SGS are very fast. Moreover, `[M3]` is sometimes faster than `[M2]` (for Δ from 4.5 to 2.5). `[M3]` also establishes the optimality for more instances, but the number of additional optimal solutions found decreases with Δ. This shows that the ISS can permit to reduce the computational time, even though it is much more complex, since several MILP are solved successively. Nonetheless, despite Proposition 5, the data generated by the repair process and propagated throughout the process is useful, until a certain point; unfortunately, when Δ becomes too small, this information becomes less and less helpful for the MILP resolution.

Table 4 Impact of instance indicators

	$\Delta = 1$			$\Delta = 2$			$\Delta = 3$			$\Delta = 4$			$\Delta = 5$		
	# Opt	*Gap (%)*	*Time (s)*	*# Opt*	*Gap (%)*	*Time (s)*	*# Opt*	*Gap (%)*	*Time (s)*	*# Opt*	*Gap (%)*	*Time (s)*	*# Opt*	*Gap (%)*	*Time (s)*
NC = 1.50 (16)	14	3.10	266	14	2.48	289	14	1.87	236	14	1.19	238	14	0.46	229
NC = 1.80 (16)	14	1.95	325	14	2.01	273	14	1.70	248	14	0.31	233	16	0.02	94
NC = 2.10 (16)	15	1.67	179	15	1.00	187	15	0.52	208	15	0.90	150	15	0.28	138
RF = 1.00 (12)	9	6.38	569	9	4.48	510	9	3.07	478	9	2.37	458	10	0.82	387
RF = 0.75 (12)	10	2.56	366	10	2.82	390	10	2.36	419	10	0.83	347	11	0.19	207
RF = 0.50 (12)	12	0.02	64	12	0.01	90	12	0.02	19	12	0.01	19	12	0.01	15
RF = 0.25 (12)	12	0.00	28	12	0.01	9	12	0.01	7	12	0.00	5	12	0.00	5
RS = 1.00 (12)	12	0.00	15	12	0.00	4	12	0.00	2	12	0.00	2	12	0.00	1
RS = 0.70 (12)	12	0.01	32	12	0.00	7	12	0.00	4	12	0.00	2	12	0.00	2
RS = 0.50 (12)	12	0.01	143	12	0.01	133	12	0.02	32	12	0.02	11	12	0.01	8
RS = 0.20 (12)	7	8.94	837	7	7.30	855	7	5.45	885	7	3.19	814	9	1.01	602

One can note that the smaller Δ, the less reduction is applied. Other strategies have been tested (e.g., linear reduction), but seem less efficient to provide pertinent information between iterations and so reduce the overall computational time.

We now provide additional insight on the performance of the best method [M3] on 48 selected instances from the J30 set w.r.t. the different values of Δ and the instance characteristics. In the PSPLIB, each instance is generated according to three indicators. The network complexity gives an indication on the density of the precedence graph. On the 48 instances, there are 16 instances with NC = 1.50 (small density), 16 instances with NC = 1.80 (medium density) and 16 instances with NC = 2.10 (high density). The resource factor gives the number of resources used by an activity from one resource (RF = 0.25) to four resources (RF = 1.00). Finally the Resource Strength measures the tightness of the resource constraints. For RS = 1.00, the earliest precedence-feasible schedule is always globally feasible (no resource constraints). For RS = 0.75, RS = 0.70, RS = 0.50 and RS = 0.20 the resource contraints are more and more tightened in the sense that less and less activities can be scheduled in parallel as their requirement is higher and higher compared to the resource availabilities. Table 4 provides for each value of Δ and for each set of instances having the same indicator, the number of optima found, the average gap at the end of the branch-and-bound and the average CPU time (limited here to 1800s). It appears that the CPU time and the gap increase asNC decreases, RF increases and RS decreases. This is inline with the previously established results for the RCPSP indicating the instances with a lower precedence graph density, a higher number of required resources and tighter resource constraints are harder to solve. Note that this is true for any value of Δ whereas for the larger value of Δ the effect is mitigated as the instances become globally easier to solve. However the instances with RS = 0.2 are still hard with only 9 out of 12 instances solved with an average CPU time of 602s.

Table 5 Average gap (%) below the optimal RCPSP makespan

	$\Delta=1$	$\Delta=2$	$\Delta=3$	$\Delta=4$	$\Delta=5$
All (48)	2.31	3.83	4.62	5.09	6.02
NC = 1.50 (16)	2.44	4.22	4.98	5.22	5.24
NC = 1.80 (16)	1.95	3.23	4.16	4.77	7.29
NC = 2.10 (16)	2.51	4.02	4.72	5.26	5.38
RF = 1.00 (12)	1.46	2.28	2.76	2.99	4.66
RF = 0.75 (12)	1.77	2.50	3.06	3.57	5.45
RF = 0.50 (12)	3.31	5.91	6.97	7.67	8.00
RF = 0.25 (12)	2.39	4.01	4.98	5.34	5.69
RS = 1.00 (12)	0.00	0.00	0.00	0.00	0.00
RS = 0.70 (12)	0.75	0.82	0.82	0.82	0.82
RS = 0.50 (12)	3.58	5.16	5.74	6.10	6.39
RS = 0.20 (12)	6.75	13.27	17.15	19.38	20.47

According to the same indicators, it is also relevant to evaluate the gain in makespan that can be obtained by aggregating the resource constraints compared to the standard RCPSP model. In practical applications where the PARCPSP model applies (Artigues et al., 2008; Paul and Knust, 2015; Haït and Artigues, 2011; Gahm et al., 2016), this will quantify the potential improvement over the standard RCPSP model. Table 5 provides

the gap below the optimal RCPSP makesan on the same 48 instances only when the optimal solution has been provably found by [M3]. It is worth remarking that the gap is already significant for $\Delta = 1$ except for the (almost) resource-unconstrained instances RS = 1.00 and RS = 0.70. Indeed for the RS = 0.20 instances, the gap goes up to 6.75% while it stays around 2% for other instances. The average over all instances shows that the gap increases in average almost by the same amount Δ is increased. The only indicator that has a visible impact on this gap is the RS indicator where the increase as Δ increases is spectacular for the hardest RS = 0.20 instances. This underlines the expected result that when resource constrained are tighter, averaging the consumption on (even small) periods is highly beneficiary for the makespan.

8 Conclusions and perspectives

In this paper, a new problem, the PARCPSP, has been introduced, that permits to tackle temporal aspects in an exact way, while the resource usage is considered on average over aggregated periods. Although it appears to be a relaxation of the RCPSP, the structure of the two problems are quite different (e.g., multiple factors of alteration of the feasibility of a schedule, no dominance rule). Both exact and heuristic methods have been proposed to solve the problem, via the definition of a MILP based on a mixed time representation, and the adaptation of standard SGS. Finally, an ISS has been presented, that enables to reduce the resolution time. The impact of the standard instance complexity indicators (network complexity, resource factor and resource strength) have been studied on the number of times optimality can be proved, on the average optimality gap and on the CPU times. This reveals that on theses indicators the PARCPSP share the same properties with the RCPSP, namely that low NC, high RF and low RS yield hard instances. However, increasing Δ significantly mitigates this behaviour. Finally the study of the gap, when optimality could be proved, with the optimal RCPSP makespan shows that on the selected instances the makespan is about $(1 + \Delta)\%$ lower for the PARCPSP than for the RCPSP. This improvement is much larger on the hard instances with low resource strength, which underlines the benefit of switching if possible to the PARCPSP model when resources are scarse.

Among future research guidelines, reinforcing the link between the continuous and discrete temporal variables in the MILP, that constitutes the model kernel, represents a daunting challenge. This could lead to significant computational results improvements.

It would also be interesting to extend the ISS to solve the RCPSP. On the one hand, the optimum gap between the RCPSP and the PARCPSP(1) may be large. On the other hand, the lower the period length, the more the number of periods; continuing decreasing the period length below one does not seem to be a reasonable option. Therefore, other alternatives should be investigated. For instance, generating cutting plane inequalities, valid for the RCPSP, into the PARCPSP model (thus defining an intermediate relaxation), could be an interesting strategy.

For further research, we will consider applications where a part of the resources are of aggregated type and a part of the resources are of detailed type, like in the employee timetabling and job shop scheduling problem considered in Artigues et al. (2008), where human resources are modelled as periodically aggregated resources, while machines are standard disjunctive resources. Multi-project problems and arbitrary subdivisions of the time horizon are also appealing research directions.

References

Allen, J.F. (1981) 'An interval-based representation of temporal knowledge', *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI*, Vol. 81, pp.221–226.

Artigues, C., Demassey, S. and Néron, E. (2008) 'Resource-constrained project scheduling: models, algorithms, extensions and applications', *Control Systems, Robotics and Manufacturing Series*, Wiley-ISTE, London.

Artigues, C., Gendreau, M., Rousseau, L-M. and Vergnaud, A. (2009) 'Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound', *Computers and Operations Research*, Vol. 36, No. 8, pp.2330–2340.

Castro, P., Harjunkoski, I. and Grossmann, I. (2009) 'New continuous-time scheduling formulation for continuous plants under variable energy cost', *Industrial Engineering Chemistry Research*, Vol. 48, No. 14, pp.6701–6714.

Castro, P., Harjunkoski, I. and Grossmann, I. (2011) 'Optimal scheduling of continuous plants with energy constraints', *Computers and Chemical Engineering*, Vol. 35, No. 2, pp.372–387.

Floudas, C. and Lin, X. (2004) 'Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review', *Computers and Chemical Engineering*, Vol. 28, No. 11, pp.2109–2129.

Gademann, N. and Schutten, M. (2005) 'Linear-programming-based heuristics for project capacity planning', *IIE Transactions*, Vol. 37, No. 2, pp.153–165.

Gahm, C., Denz, F., Dirr, M. and Tuma, A. (2016) 'Energy-efficient scheduling in manufacturing companies: a review and research framework', *European Journal of Operational Research*, Vol. 248, No. 3, pp.744–757.

Haït, A. and Artigues, C. (2011) 'A hybrid CP/MILP method for scheduling with energy costs', *European Journal of Industrial Engineering*, Vol. 5, No. 4, pp.471–489.

Haït, A. and Baydoun, G. (2012) 'A new event-based MILP model for the resource-constrained project scheduling problem with variable intensity activities (RCPSVP)', *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management*, IEEM, Hong Kong.

Hans, E.W. (2001) *Resource Loading by Branch-and-Price Techniques*, Twente Univ. Press, Enschede, 151pp.

Kis, T. (2005) 'A branch-and-cut algorithm for scheduling of projects with variable-intensity activities', *Mathematical Programming*, Vol. 103, No. 3, pp.515–539.

Kolisch, R. and Sprecher, A. (1996) 'PSPLIB – a project scheduling problem library', *European Journal of Operational Research*, Vol. 96, No. 1, pp.205–216.

Mestry, S., Damodaran, P. and Chen, C-S. (2011) 'A branch and price solution approach for order acceptance and capacity planning in make-to-order operations', *European Journal of Operational Research*, Vol. 211, No. 3, pp.480–495.

Morin, P-A., Artigues, C. and Haït, A. (2016) 'A new relaxation of the resource-constrained project scheduling problem', *Proceedings of the 15th International Conference on Project Management and Scheduling*, PMS, Valencia, Spain.

Paul, M. and Knust, S. (2015) 'A classification scheme for integrated staff rostering and scheduling problems', *RAIRO-Operations Research*, Vol. 49, No. 2, pp.393–412.

Pinedo, M.L. (2016) *Scheduling: Theory, Algorithms and Systems*, Springer International Publishing, Cham.

Schwindt, C. and Zimmermann, J. (2015) 'Handbook on project management and scheduling', *International Handbooks on Information Systems*, Springer, Berlin Heidelberg, Vol. 1.

Silvente, J., Aguirre, A., Zamarripa, M., Méndez, C., Graells, M. and Espuña, A. (2015) 'Improved time representation model for the simultaneous energy supply and demand management in microgrids', *Energy*, Vol. 87, pp.615–627.