A rough-cut capacity planning model with overlapping

Georges Baydoun[1] · Alain Haït[2] · Robert Pellerin[1] · Bernard Clément[1] · Guillaume Bouvignies[2]

Abstract In the early phases of projects, capacity planning is performed to assess the feasibility of the project in terms of delivery date, resource usage and cost. This tactical approach relies on an aggregated representation of tasks in work packages. At this level, aggressive project duration objectives are achieved by adopting work package overlapping policies that affect both workload and resource usage. In this article, we propose a mixed-time MILP model for project capacity planning with different possibilities for overlapping levels between work packages. In the model, the planning time horizon is divided into time buckets used to evaluate resource usage, while starting and ending times for work packages are continuous. The model was tested on a benchmark of 5 sets of 450 theoretical instances each. More than half of the tested instances were solved to optimality within 500 s. Results also show that, while overlapping is more beneficial for accelerating project delivery times, it can still have a positive impact on project cost by allowing a better distribution of workload.

✉ Georges Baydoun
georges.baydoun@polymtl.ca

Alain Haït
alain.hait@isae.fr

Robert Pellerin
robert.pellerin@polymtl.ca

Bernard Clément
bernard.clement@polymtl.ca

Guillaume Bouvignies
guillaume.bouvignies@isae.fr

1 Department of Mathematics and Industrial Engineering, École Polytechnique, Montreal, Canada

2 Institut Supérieur de l'Aéronautique et de l'Espace-Supaero, University of Toulouse, Toulouse, France

Finally, overlapping options seem to have less influence on the performance of the model than project slack or number of work packages.

Keywords Rough-cut capacity planning · Concurrent engineering · Overlapping

1 Introduction

Overlapping is a common practice used in construction and in product development projects to accelerate the execution of large projects. This technique consists of executing in parallel two sequential activities by allowing a downstream activity to start before the end of an upstream activity based on preliminary information. However, the overlapping of activities can entail rework tasks and modifications as a result of the transmission of complementary information after the start of the downstream activity (Berthaut et al. 2014). Activity overlapping thus allows the total duration of the project execution to be reduced at the expense of additional workload and execution cost associated with rework.

Several authors have studied the relation between rework and the amount of overlap in a project conducted in a concurrent engineering context or fast-tracking mode. For instance, Gerk and Qassim (2008) proposed a linear model for project acceleration using crashing, overlapping and activity substitution. Activity crashing includes the allocation of additional resources for an activity in order to accelerate its execution. Activity substitution is another technique for accelerating projects by replacing one (or more) activity with another activity.

Berthaut et al. (2014) and Grèze et al. (2012) proposed a more realistic approach by restricting overlapping possibilities to a set of feasible overlap durations for each couple of overlappable activities, instead of considering a continuous and linear relation between overlap amount and rework. This assumption is more realistic as overlapping points between activities are defined through clear document or information exchange in a concurrent engineering context, which limits the overlapping modes to a reduced and discrete set of possibilities.

However, these resource-constrained project scheduling problem (RCPSP) models are not suited for planners in the early phases of projects, as detailed activity content, duration, and resources are not precisely known, and as work intensity cannot be assumed to be constant over execution time. Indeed, planners tend to adopt an aggregate planning approach in large engineering projects where work packages (WPs) are broadly defined as groups of multiple activities that could extend over a long period (i.e. weeks or months) (Cherkaoui et al. 2013). In practice, project planning is done by preparing several schedules at different phases of the project, where aggregation levels depend on the ongoing phase and on the target audience. For instance, the front-end-loading (FEL) approach, commonly used in large construction projects, is composed of successive planning stages. Early phases involve tactical planning based on rough-cut capacity planning (RCCP) techniques in order to fix all project milestones and estimate resource usage. At this level, projects are divided into work packages (WPs) which are clusters of activities. Rough-cut capacity planning (RCCP) models divide the planning horizon into time buckets (or periods) used to evaluate critical resource

usage and by allowing resource allocation for WP to vary from one period to another (De Boer 1998).

Recognizing the need of practitioners to better support the project planning function in the early phases of projects, this paper proposes an exact RCCP model that determines the order of execution in a set of WPs so as to minimize the total project duration and/or project cost, while respecting precedence relations, resource constraints and taking into consideration overlapping possibilities. The proposed model considers variable WP intensities and aggregate resource capacities.

The remainder of the paper is organized as follows. We first give a brief state of the art of existing RCCP models and overlapping models in Sect. 2. We then introduce the original mixed-time RCCP model in Sect. 3, before explaining our new RCCP model with multiple overlapping modes in Sect. 4. Section 5 explains the generation of our test instances, and an illustrative example is presented in Sect. 6. Finally, Sect. 7 analyzes the performance and the results of our model, before some concluding remarks are provided in Sect. 8.

2 Related work

In the order acceptance stage of a project, companies tend to commit to due dates without accurate knowledge of their resource capacities. The RCCP ensures, at an aggregated level, that the capacities of critical resources are sufficient to complete a project within its time and cost limits (De Boer 1998). Performed at the tactical level, the RCCP is based on a horizon divided into time buckets (or periods) used to evaluate the resource usage. The WPs are defined by their work content, and resource allocation can vary from one period to another. Capacity and resource allocation flexibility allow the WP durations to be adapted according to time and cost-related considerations. WPs may start or end during a period; therefore, it is possible to plan a WP and its successor within the same period.

Aside from denominated RCCP models, RCPSP models, where intensities can vary from period to period, are also suitable for the RCCP problem as they consider fixed workload for each activity and variable resource usage between periods, and therefore variable activity durations. RCPSP models that are suitable for the RCCP have several names in the literature such as resource constrained project scheduling with variable intensity activities (RCPSVP) and RCPSP with flexible resource profiles.

Wullink (2005) distinguishes three classes of solution approaches for the RCCP: straightforward constructive heuristics, LP based heuristics and exact approaches. Among existing RCCP models, Hans (2001) proposed an exact approach that consists of determining the periods where each job can be executed, and then specifying the fractions of the WP contents that are actually executed in each period. However, this method can allow a predecessor and any direct successor to be performed in the same period without determining their ending and starting times within the period, which could lead to precedence infeasibility. Hans (2001) proposed two alternatives to avoid this problem by over-constraining the problem. Taking a project scheduling point of view, Kis (2005) proposed an RCPSP model with variable activity intensities, which

forbids two activities with a direct precedence relation to be executed in the same period.

More recently, Haït and Baydoun (2012) proposed an exact approach that consists of using continuous time representation for events, together with a discrete evaluation of resources. This means that start and ending dates of WPs can be determined within a time period, making it possible to guarantee that precedence constraints will be respected. Yet, resource consumptions are still evaluated globally over periods, making the model suitable for planning at a tactical level where planning is performed in practice on a period-by-period basis (i.e. resource availabilities and allocations are determined per period).

Recently, Naber and Kolisch (2014) addressed the RCPSP with flexible resource profiles, meaning that resource usage of an activity can vary from period to period. Their objective is to minimize project makespan while respecting availabilities per periods of all resources. They propose four different discrete-time MILP model formulations based on existing formulations and compare their performance. Their experiments show that the model called "FP-DT3", that is based on the RCPSP formulation of Bianco and Caramia (2013) and the RCPSVP model of Kis (2005), dominates the other formulations in both solution quality and run-time. Several heuristic approaches have also been proposed to solve the RCCP problem including constructive heuristics (De Boer 1998) and linear-programming based heuristics (Gademann and Schutten 2005).

Although some authors proposed interesting extensions to the precedence relations of RCPSVP models by allowing overlapping, none of these models considers overlapping and rework. In fact, Kis (2006) extended his RCPSVP model by introducing feeding precedence constraints: a successor can start after a percentage of the execution of his direct predecessor is completed. Alfieri et al. (2011) extended feeding precedence constraints to generalized precedence relations. However, both Kis (2006) and Alfieri et al. (2011) do not take reworks into account. Moreover, the models of Kis (2006) and Alfieri et al. (2011) have only one possibility of precedence constraints and do not consider several feasible modes of overlapping with different amounts of rework.

Despite these recent advances, models that consider feeding precedence constraints do not allow the overlapping of WPs that are normally executed consecutively. This concurrent engineering method is a trend in product development and is also widely used in contexts such as construction in order to accelerate project execution. It also adds flexibility in starting and ending times, allowing for better use of regular capacities and thus reducing the need for external resources. However, overlapping adds workloads, "reworks", on both down-stream and up-stream WPs (Berthaut et al. 2014). Reworks are a result of information exchange and eventual alteration of the work, which are caused by starting a down-stream WP before all finalized information is available from up-stream WP.

To fill this gap, we propose a mixed integer linear-programming model, which is an extension of the RCCP model proposed by Haït and Baydoun (2012), where predecessor–successor WPs can overlap according to multiple overlapping modes. This extended model assumes that each overlapping mode can be defined by the percentage of execution of a predecessor WP that needs to be reached in order to start

the successor, as well as the amount of reworks on both WPs. The model is further explained in the following section.

3 Mixed-time RCCP model

This section presents the original model by Haït and Baydoun (2012). It combines a continuous time representation of the starting and ending dates of WPs and a discrete time evaluation of resources. A set of binary variables ensures the relation between starting time, ending time and durations over periods. These durations give minimum and maximum workload that can be assigned to the period.

In this section only, the starting and ending times of WP i are denoted by continuous variables t_i^0 and t_i^1, respectively. Binary variables z_{ip}^0 and z_{ip}^1 equal 1 if the starting and ending times (respectively) of WP i occur during or before time period p. Variables d_{ip} and l_{ip} are respectively the duration and the assigned workload of WP i over the period p. In Sect. 4, the aforementioned notations will be generalized in order to handle overlapping. Other definitions that are relevant to this section are given in Table 1.

In addition to basic definition-related constraints, there are four types of constraints for the model by Haït and Baydoun (2012): those ensuring the link between continuous and binary variables for starting and ending times, the ones concerning the durations over periods, scheduling constraints, and finally, workload and intensity constraints.

3.1 Continuous/binary variables constraints

The following constraints (1)–(3) ensure that binary variables z_{ip}^0 and z_{ip}^1 equal 1 only in the periods during or after starting and ending events (respectively). These binary variables were first introduced by Pritsker and Watters (1968) as a step formulation for scheduling with limited resources, and used by several other authors including Bianco and Caramia (2013). However, only the model by Haït and Baydoun (2012) used them

Table 1 Nomenclature: sets and parameters

Set/parameter	Description
P, D, H	Set of time periods ($p \in P$), duration of a period, time horizon $H = D.\|P\|$. Period p starts at time $D(p-1)$ and finishes at $D.p$
I	Set of work packages ($i \in I$)
$\mathrm{RD}_i, \mathrm{DD}_i$	Release date and due date of WP i
Succ_i	Set of successors of i (Finish-to-Start, without overlapping)
L_i	Total required workload for WP i
$B_i^{\min}, B_i^{\max}$	Minimum and maximum workload that can be assigned for i during a period of duration D
R	Set of resources ($r \in R$)
K_{rp}	Available regular capacity of resource r during period p
A_{ir}	Fraction of the workload of i provided by resource r
C_r^{ext}	Unitary cost of extra-capacity for resource r

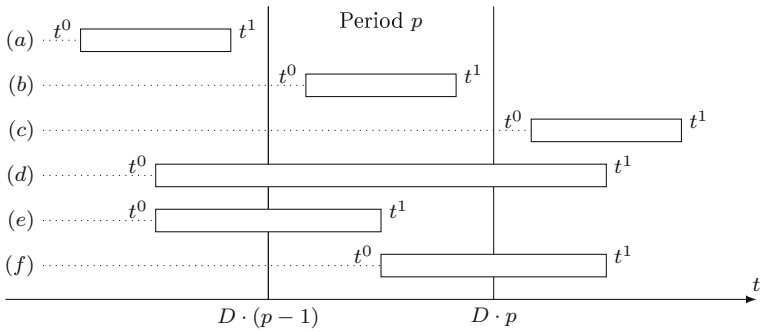

Fig. 1 The six possible configurations for a WP regarding a time period p

in the context of a mixed continuous and discrete time representation.

$$t_i^c \geq D \cdot p \cdot \left(1 - z_{ip}^c\right) \qquad \forall i \in I, c \in \{0,1\}, p \in P \tag{1}$$

$$t_i^c \leq D \cdot p + H \cdot \left(1 - z_{ip}^c\right) \qquad \forall i \in I, c \in \{0,1\}, p \in P \tag{2}$$

$$z_{ip+1}^c \geq z_{ip}^c \qquad \forall i \in I, c \in \{0,1\}, p \in \{1 \ldots |P| - 1\} \tag{3}$$

3.2 Durations over periods

When considering a WP i and a time period p, starting and ending events can be before, during, or after p. Therefore, each couple WP i and time period p has six possible configurations, depicted in Fig. 1.

The following constraints give the relations between WP durations over periods (denoted d_{ip}) on one side, binary variables z_{ip}^0 and z_{ip}^1 and continuous time variables t_i^0 and t_i^1 on the other side:

$$d_{ip} \leq D \cdot \left(z_{ip}^0 - z_{ip-1}^1\right) \qquad \forall i \in I, p \in P \tag{4}$$

$$d_{ip} \geq D \cdot \left(z_{ip-1}^0 - z_{ip}^1\right) \qquad \forall i \in I, p \in P \tag{5}$$

$$d_{ip} \geq t_i^1 - D \cdot p + D \cdot z_{ip-1}^0 - H \cdot \left(1 - z_{ip}^1\right) \qquad \forall i \in I, p \in P \tag{6}$$

$$d_{ip} \geq D \cdot p \cdot \left(1 - z_{ip-1}^0\right) - t_i^0 - D \cdot z_{ip}^1 \qquad \forall i \in I, p \in P \tag{7}$$

$$\sum_{p=1}^{|P|} d_{ip} = t_i^1 - t_i^0 \qquad \forall i \in I \tag{8}$$

Constraints (4) force d_{ip} to 0 when WP i is not active during period p [configurations (a) and (c)] and limit its value to period duration D if i and p are in any of the four other configurations. Inequalities (5), (6), and (7) concern configurations (d), (e), and (f) respectively. Inequalities (5) give d_{ip} lower-bounds if WP i begins before period p

and is not completed during p. Constraints (6) [respectively (7)] provide lower-bounds for d_{ip} when WP i is finished (respectively started) in period p. Finally, constraints (8) provide global coherence between starting and ending times, and durations per periods.

3.3 Scheduling constraints

Each WP has its own time window, consisting of a release date RD_i and due date DD_i. Constraints (9) and (10) ensure that WP i is executed in its allowed time window. Moreover, a successor cannot start before its predecessor is completed [constraints (11)].

$$t_i^0 \geq \mathrm{RD}_i \qquad \forall i \in I \tag{9}$$

$$t_i^1 \leq \mathrm{DD}_i \qquad \forall i \in I \tag{10}$$

$$t_j^0 \geq t_i^1 \qquad \forall i \in I,\ j \in \mathrm{Succ}_i \tag{11}$$

3.4 Workload and intensity constraints

Intensity constraints ensure that the part of a WP's workload assigned to a period, l_{ip}, respects minimum and maximum allowed intensities $B_i^{\min}$ and $B_i^{\max}$ [constraints (12) and (13)]. Workload constraints guarantee that the total assigned workload over the time horizon matches the required workload for the WP [equalities (14)].

$$l_{ip} \geq B_i^{\min} \cdot \frac{d_{ip}}{D} \qquad \forall i \in I,\ p \in P \tag{12}$$

$$l_{ip} \leq B_i^{\max} \cdot \frac{d_{ip}}{D} \qquad \forall i \in I,\ p \in P \tag{13}$$

$$\sum_{p \in P} l_{ip} = L_i \qquad \forall i \in I \tag{14}$$

3.5 Definition-related constraints

The following constraints are straight-forward results of definitions of makespan, regular and extra resource allocations y_{rp} and y_{rp}^{ext}, and cost. Variables y_{rp} and y_{rp}^{ext} are respectively regular and non-regular allocations for resource r during period p. While we dispose of K_{rp} regular capacity for resource r during period p [constraints (17)], extra capacities come at an additional price. Project cost is calculated as the total cost of extra capacities used for the whole project [constraint (18)].

$$\text{makespan} \geq t_i^1 \qquad \forall i \in I \tag{15}$$

$$y_{rp} + y_{rp}^{\mathrm{ext}} \geq \sum_{i \in I} A_{ir} \cdot l_{ip} \qquad \forall r \in R,\ p \in P \tag{16}$$

$$y_{rp} \leq K_{rp} \qquad \forall r \in R,\ p \in P \tag{17}$$

$$\text{cost} = \sum_{r \in R} \sum_{p \in P} C_r^{\text{ext}} . y_{rp}^{\text{ext}} \tag{18}$$

Possible objective functions are project makespan, project cost, or a trade-off between makespan and cost.

4 Mixed-time RCCP model with overlapping

Overlapping is a trend in construction projects in order to accelerate project execution. The mixed-time RCCP model only considers Finish-to-Start precedence relations between WPs and is inadequate to create tactical project plans with overlapping between WPs. To better support project planners, we propose an extension to the mixed-time RCCP model in order to allow overlapping.

4.1 Overlapping work packages

Our new model is based on the same representation of time and events of WP start and WP end, but adds a third type of events: intermediate milestones. These milestones reflect the attainment of a certain development state of a WP. They may correspond to important decision making moments, or the completion of key deliverables.

Figure 2 presents an example of a WP A with one successor. WP B can start after the milestone t_A^1, overlapping its predecessor. WP A is divided into two parts by this milestone and the ending event of A is now denoted t_A^2. The position of t_A^1 depends on the workload repartition from the beginning of A.

More generally, a WP i may have several overlapping milestones (one for each successor that may overlap). We denote by C_i the number of these milestones that cut the work package. WP i is therefore divided into $C_i + 1$ parts. The milestones are denoted $t_i^1 \ldots t_i^{C_i}$ and $t_i^{C_i+1}$ is the end of the WP. Each part c has its own set of

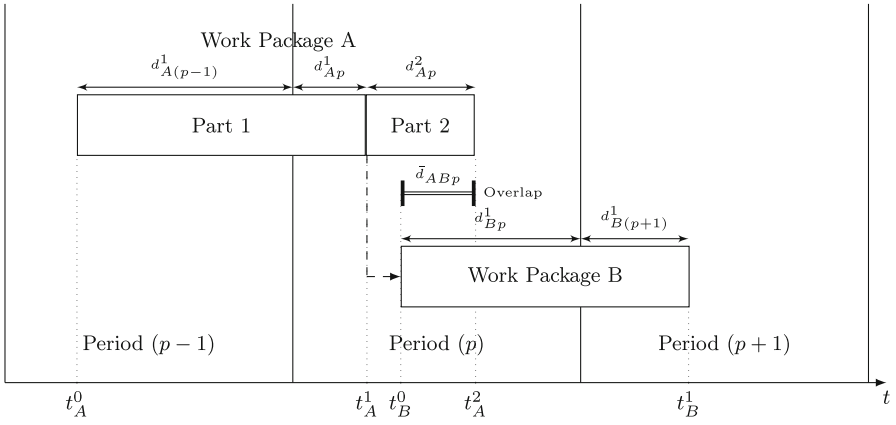

Fig. 2 Work package A with one successor B that can overlap

durations over the periods, denoted d_{ip}^{c}, and assigned workload during the periods, l_{ip}^{c}. We also define overlapping durations over periods $\bar{d}_{ijp}$ (e.g. $\bar{d}_{ABp}$ in Fig. 2).

Finally, as explained in Sect. 1, overlapping involves a certain quantity of rework that affects the successor as well as the predecessor. This additional work is performed during the overlap and will be represented by variables $\bar{l}_{ijp}^{\text{pred}}$ and $\bar{l}_{ijp}^{\text{succ}}$ for the predecessor and the successor, respectively.

4.2 Overlapping modes

Take again the example of Fig. 2. The new model has to answer the question: "how to overlap ?" by choosing among several positions of milestone t_A^1. For example, it could be possible to start B after 60 % of the workload of A is complete, with a certain amount of rework. B could also start after 80 % of A, less interesting in terms of project duration but with fewer rework. Finally, B could start after the end of A, with no rework. Hence we have to choose among three overlapping modes between A and B. Note that there is only one milestone t_A^1 and its position will change according to the selected mode.

Consider now the case of a WP A and two successors, B and C, that may overlap. WP A is therefore divided into three parts ($C_A = 2$). WP B can start after 70 or 100 % (no overlapping) of A is complete in terms of workload. WP C can start after 35 and 100 % of the total workload of A. These alternatives involve four overlapping modes for A with its successors as depicted in Fig. 3. The relative position of the milestones associated with B and C depends on the selected mode.

In the model, the selection of a mode m for WP i will be performed by binary variables e_{im}. For each mode, the position of the milestone associated to each successor is given by parameter pos_{ijm}. We also define a binary parameter ov_{ijm} to identify the case of no overlap between i and j in mode m.

Tables 2 and 3 give the nomenclature for the new RCCP model with overlapping. The extension of the RCCP model to overlapping between work packages is given by constraints (19)–(50) described in the following subsections.

4.3 Continuous/binary variables constraints

Constraints (19), (20), and (21) are straight-forward results of the definition of z_{ip}^{c}, and generalize constraints (1)–(3).

$$t_i^c \geq D \cdot p \cdot \left(1 - z_{ip}^c\right) \qquad \forall i \in I, c \in \{0..C_i + 1\}, p \in P \tag{19}$$

$$t_i^c \leq D \cdot p + H \cdot \left(1 - z_{ip}^c\right) \qquad \forall i \in I, c \in \{0..C_i + 1\}, p \in P \tag{20}$$

$$z_{ip+1}^c \geq z_{ip}^c \qquad \forall i \in I, c \in \{0..C_i + 1\}, p \in \{1 \ldots |P| - 1\} \tag{21}$$

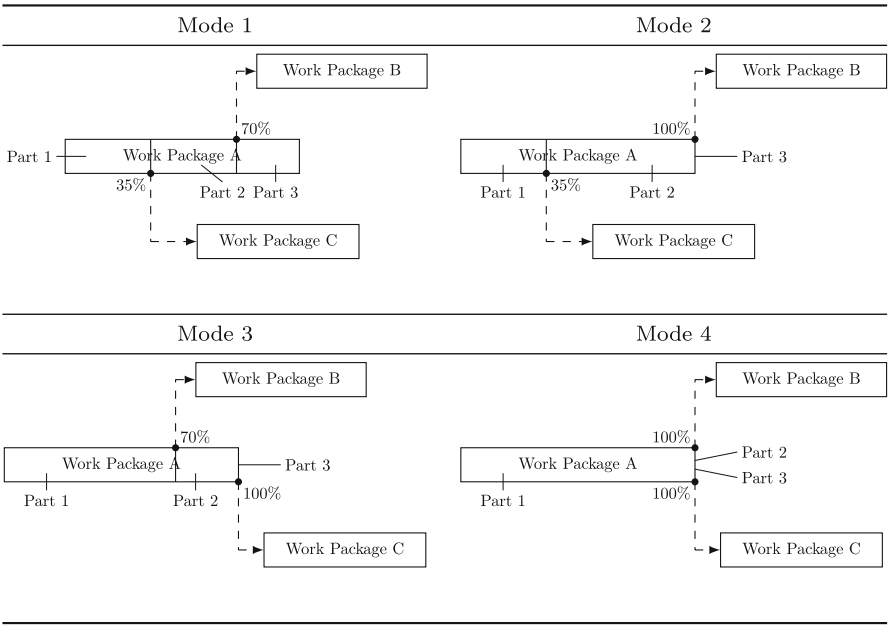

Fig. 3 Four possible overlapping modes for a WP A with two successors B and C that can overlap

Table 2 Nomenclature: additional sets and parameters for the case with overlapping

Set/parameter	Description
$\overline{\text{Pred}_i}$	Set of predecessors of WP i that can overlap on i
$\overline{\text{Succ}_i}$	Set of successors of WP i that can overlap on i
C_i	Number of successor of WP i that can overlap on i
M_i	Set of overlapping modes between i and its direct successors ($m \in M_i$)
pos_{ijm}	Position of j among the successors of i according to mode $m \in M_i$
ov_{ijm}	Binary parameter that equals 1 if WPs i and j overlap in mode $m \in M_i$, 0 otherwise
L^c_{im}	Required workload of part c of WP i in mode m
$\bar{L}^{\text{pred}}_{ijm}$	Required rework on WP i in mode $m \in M_i$ due to overlapping between i and j
$\bar{L}^{\text{succ}}_{ijm}$	Required rework on WP j in mode $m \in M_i$ due to overlapping between i and j

4.4 Durations over periods

Constraints (22)–(26) provide the link between durations per periods d^c_{ip} and variables t^c_i and z^c_{ip}, similar to Eqs. (4)–(8) of the original mixed-time model.

Table 3 Nomenclature: new variables for the case with overlapping

Variable	Description
$t_i^0, t_i^{C_i+1}$	Starting time and ending time of i
t_i^c	For $c \in \{1 \dots C_i\}$: ending time of part c of i
z_{ip}^c	For $c \in \{0 \dots C_i + 1\}$: binary variable that equals 1 if t_i^c is in period p or before
e_{im}	Binary variable that equals 1 if mode m is chosen for i, 0 otherwise
d_{ip}^c	Duration of part c of WP i within period p
l_{ip}^c	Workload of part c of WP i during period p
$\bar{d}_{ijp}$	Duration of overlapping between WPs i and j within period p
$\bar{l}_{ijp}^{\text{pred}}$	Rework on WP i during p due to overlapping between WPs i and j
$\bar{l}_{ijp}^{\text{succ}}$	Rework on WP j during p due to overlapping between WPs i and j

$$d_{ip}^c \leq D \cdot \left(z_{ip}^{c-1} - z_{ip-1}^c\right) \qquad \forall i \in I, c \in \{1 \dots C_i + 1\}, p \in P \tag{22}$$

$$d_{ip}^c \geq D \cdot \left(z_{ip-1}^{c-1} - z_{ip}^c\right) \qquad \forall i \in I, c \in \{1 \dots C_i + 1\}, p \in P \tag{23}$$

$$d_{ip}^c \geq t_i^c - D \cdot p + D \cdot z_{ip-1}^{c-1} - H \cdot \left(1 - z_{ip}^c\right) \qquad \forall i \in I, c \in \{1 \dots C_i + 1\}, p \in P \tag{24}$$

$$d_{ip}^c \geq D \cdot p \cdot \left(1 - z_{ip-1}^{c-1}\right) - t_i^{c-1} - D \cdot z_{ip}^c \qquad \forall i \in I, c \in \{1 \dots C_i + 1\}, p \in P \tag{25}$$

$$\sum_{p=1}^{|P|} d_{ip}^c = t_i^c - t_i^{c-1} \qquad \forall i \in I, c \in \{1 \dots C_i + 1\} \tag{26}$$

Constraints (27)–(29) concern overlapping durations within periods by giving upper-bounds for $\bar{d}_{ijp}$. Constraints (27) and (28) make sure that overlapping durations within periods never exceed durations within periods of both predecessor i and successor j. Constraints (29) ensure that overlapping durations do not surpass the span between the starting time of j and ending time of i. Note that inequalities (27) consider the assumption that for a given WP, its overlapping predecessors can never overlap on its overlapping successors (no cascade effect).

$$\bar{d}_{ijp} \leq d_{jp}^1 \qquad \forall i \in I, j \in \overline{\text{Succ}_i}, p \in P \tag{27}$$

$$\bar{d}_{ijp} \leq \sum_{c=1}^{C_i+1} d_{ip}^c \qquad \forall i \in I, j \in \overline{\text{Succ}_i}, p \in P \tag{28}$$

$$\bar{d}_{ijp} \leq t_i^{C_i+1} - t_j^0 + H \cdot (1 - e_{im}) \quad \forall i \in I,\ j \in \overline{\text{Succ}_i},\ m \in \{M_i | \text{ov}_{ijm} = 1\},\ p \in P \tag{29}$$

4.5 Scheduling constraints

Constraints (30) and (32) force WP i to be in its allowed time window, while constraints (31) make sure that the different parts of WP i are in order. Moreover, the end of each part gives the time t_i^c when a successor in $\overline{\text{Succ}_i}$ can start. Constraints (33) ensure that successor j begins after the correct milestone of WP i. For successors in Succ_i that cannot overlap on i, a classical end-to-start constraint is defined in constraints (34). For successors in $\overline{\text{Succ}_i}$ that can overlap on i, constraints (35) exclude the case in which a predecessor of i and a successor of i overlap (no cascade effect).

$$t_i^0 \geq \text{RD}_i \qquad \forall i \in I \tag{30}$$

$$t_i^{c+1} \geq t_i^c \qquad \forall i \in I, c \in \{0 \dots C_i\} \tag{31}$$

$$t_i^{C_i+1} \leq \text{DD}_i \qquad \forall i \in I \tag{32}$$

$$t_j^0 \geq t_i^c - H \cdot (1 - e_{im}) \qquad \forall i \in I, m \in M_i,\ j \in \overline{\text{Succ}_i},\ c \in \{1 \dots C_i + 1 | c = \text{pos}_{ijm}\} \tag{33}$$

$$t_j^0 \geq t_i^{C_i+1} \qquad \forall i \in I,\ j \in \text{Succ}_i \tag{34}$$

$$t_j^1 \geq t_i^{C_i+1} \qquad \forall i \in I,\ j \in \overline{\text{Succ}_i} \tag{35}$$

4.6 Workload and intensity constraints

Constraints (36) and (37) ensure that the required workload is attained for each part of WP i according to the selected mode.

$$\sum_{p \in P} l_{ip}^c \geq L_{im}^c \cdot e_{im} \qquad \forall i \in I, c \in \{1 \dots C_i + 1\}, m \in M_i \tag{36}$$

$$\sum_{p \in P, c \in \{1..C_i+1\}} l_{ip}^c = L_i \qquad \forall i \in I \tag{37}$$

In addition to initial workloads, reworks are required on predecessors and successors that overlap. The reworks should be executed during the overlapping time between predecessors and successors. Constraints (38) and (39) make sure that the total executed reworks match the required reworks in selected modes.

$$\sum_{p \in P} \bar{l}_{ijp}^{\text{pred}} \geq \bar{L}_{ijm}^{\text{pred}} \cdot e_{im} \qquad \forall i \in I, m \in M_i,\ j \in \overline{\text{Succ}_i} \tag{38}$$

$$\sum_{p \in P} \bar{l}_{ijp}^{\text{succ}} \geq \bar{L}_{ijm}^{\text{succ}} \cdot e_{im} \qquad \forall i \in I, m \in M_i,\ j \in \overline{\text{Succ}_i} \tag{39}$$

Workload variables l_{ip}^{c}, $\bar{l}_{ijp}^{\text{pred}}$ and $\bar{l}_{ijp}^{\text{succ}}$ are linked to durations d_{ip}^{c} and $\bar{d}_{ijp}$ to ensure the respect of minimum and maximum allowed intensities. Constraints (40) and (41) guarantee that total assigned workload for WP i in period p stays in the allowed workload window for the total duration d_{ip}^{c} in period p.

$$\sum_{c=1}^{C_i+1} l_{ip}^{c} + \sum_{j \in \overline{\text{Succ}_i}} \bar{l}_{ijp}^{\text{pred}} + \sum_{k \in \overline{\text{Pred}_i}} \bar{l}_{kip}^{\text{succ}} \leq B_i^{\max} \cdot \frac{\sum_{c=1}^{C_i+1} d_{ip}^{c}}{D} \qquad \forall i \in I, p \in P \tag{40}$$

$$\sum_{c=1}^{C_i+1} l_{ip}^{c} + \sum_{j \in \overline{\text{Succ}_i}} \bar{l}_{ijp}^{\text{pred}} + \sum_{k \in \overline{\text{Pred}_i}} \bar{l}_{kip}^{\text{succ}} \geq B_i^{min} \cdot \frac{\sum_{c=1}^{C_i+1} d_{ip}^{c}}{D} \qquad \forall i \in I, p \in P \tag{41}$$

Constraints (42) and (43) make sure that maximum and minimum intensities are respected for initial workload for every part of a WP, while constraints (44) and (45) ensure that maximum intensity is respected for reworks.

$$l_{ip}^{c} \leq B_i^{\max} \cdot \frac{d_{ip}^{c}}{D} \qquad \forall i \in I, c \in \{1 \ldots C_i + 1\}, p \in P \tag{42}$$

$$l_{ip}^{c} \geq B_i^{\min} \cdot \frac{d_{ip}^{c}}{D} \qquad \forall i \in I, c \in \{1 \ldots C_i + 1\}, p \in P \tag{43}$$

$$\bar{l}_{ijp}^{\text{pred}} \leq B_i^{\max} \cdot \frac{\bar{d}_{ijp}}{D} \qquad \forall i \in I, j \in \overline{\text{Succ}_i}, p \in P \tag{44}$$

$$\bar{l}_{ijp}^{\text{succ}} \leq B_j^{\max} \cdot \frac{\bar{d}_{ijp}}{D} \qquad \forall i \in I, j \in \overline{\text{Succ}_i}, p \in P \tag{45}$$

4.7 Definition-related constraints

The following constraints are straight-forward results of the definition of variables e_{im}, y_{rp}^{int}, y_{rp}^{ext}, makespan, and cost.

$$\sum_{m \in M_i} e_{im} = 1 \qquad \forall i \in I \tag{46}$$

$$\text{makespan} \geq t_i^{C_i+1} \qquad \forall i \in I \tag{47}$$

$$y_{rp} + y_{rp}^{\text{ext}} \geq \sum_{i \in I} A_{ir} \cdot \left(\sum_{c=1}^{C_i+1} l_{ip}^{c} + \sum_{j \in \overline{\text{Succ}_i}} \bar{l}_{ijp}^{\text{pred}} + \sum_{k \in \overline{\text{Pred}_i}} \bar{l}_{kip}^{\text{succ}} \right) \qquad \forall r \in R, p \in P \tag{48}$$

Table 4 New parameters for the modified instances generation

Parameter	Description
ω	Fraction of predecessor–successor couples that can overlap. $0 \leq \omega \leq 1$
$[M^{\min}, M^{\max}]$	Range of overlapping modes for every predecessor–successor couple that can overlap. $M^{\min} \geq 1$
$\mathrm{Ov}_{\%}$	Percentage that characterizes the amount of overlapping
$\alpha^{\mathrm{pred}}, \alpha^{\mathrm{succ}}$	Coefficients that characterize the quantity of needed rework on predecessors and successors. $0 \leq \alpha \leq 1$

$$y_{rp} \leq K_{rp} \qquad \forall r \in R,\ p \in P \tag{49}$$

$$\text{cost} = \sum_{r \in R} \sum_{p \in P} C_r^{\text{ext}} . y_{rp}^{\text{ext}} \tag{50}$$

Possible objective functions are project makespan, project cost, or a trade-off between makespan and cost.

5 Instances generation

5.1 Original test instances

In order to test our model, we used a set of 450 instances that were generated by De Boer (1998) and that are commonly used to test RCCP models. Each instance consists of one project and was generated randomly by applying a procedure developed by Kolisch et al. (1995).

Using three parameters, De Boer (1998) generated 45 classes of 10 instances each. A class is characterized by the number of WP n, the total number of resources r, and its average slack s. The latter parameter is defined as follows: $s = \frac{\sum_{j \in J} \mathrm{DD}_j - \mathrm{RD}_j - D_{\min_j} + 1}{n}$, where $D_{\min_j}$ is the minimum duration of WP j.

De Boer's 450 instances have been generated with three different values for parameter n (10, 20, or 50 activities), three for r (3, 10, or 20 resources) and five for s (2, 5, 10, 15 and 20). The following section explains how we modified these instances so as to allow overlapping.

5.2 Modified test instances

In addition to the previous parameters n, r, and s, we introduce six more parameters for our modified instances generation. These parameters are defined in Table 4.

Parameter ω is based on the number of predecessor–successor couples in the original instance, i.e. the couples of work packages linked by a direct precedence constraint. By definition, ω represents the ratio between the number of couples that may overlap and the total number of predecessor–successor couples (denoted Total Number Of Couples) in the instance. If $\omega = 0$, then there is no overlap: this corresponds to the original instances. In order to generate modified instances for a given value of ω greater than 0, we randomly choose $\lceil \omega \cdot \text{Total Number Of Couples} \rceil$ couples to overlap.

Suppose that a couple (i, j) can overlap. Then its number of overlapping modes, denoted M_{ij}, is randomly chosen between $M^{\min}$ and $M^{\max}$. The milestone that authorizes the beginning of successor j is associated, for each mode, to a percentage of L_i, the total required workload of the predecessor. In our modified instances, this percentage is given by

$$(100\ \% - (M_{ij} - m) \cdot \text{Ov}_{\%}) \qquad \forall m \in \{1 \ldots M_{ij}\}. \tag{51}$$

Finally, for each mode, the needed rework is calculated as follows:

$$\text{Predecessor: } \left((M_{ij} - m) \cdot \frac{\text{Ov}_{\%}}{100}\right) \alpha^{\text{pred}} \cdot L_i \qquad \forall m \in \{1 \ldots M_{ij}\} \tag{52}$$

$$\text{Successor: } \left((M_{ij} - m) \cdot \frac{\text{Ov}_{\%}}{100}\right) \alpha^{\text{succ}} \cdot L_i \qquad \forall m \in \{1 \ldots M_{ij}\} \tag{53}$$

For the remainder of this paper, we fixed $M^{\min} = 2$, $M^{\max} = 3$, $\text{Ov}_{\%} = 20\ \%$, and $\alpha^{\text{pred}} = \alpha^{\text{succ}} = 0.4$ for all generated instances, and only varied parameter ω.

In the last step we recalculated the release and due dates of the WPs. In fact, time windows were tightened in the original instances, considering simple Finish-to-Start precedence relations (De Boer 1998). This makes them inadequate for our instances that allow overlapping. We explain in detail our calculations of the release and due dates for our modified instances in Sect. 5.3.

5.3 Release and due dates

In the instances of De Boer (1998), time windows were first calculated using longest path calculations. They were then tightened in order to respect a maximum slack and an average slack. After each modification, all release and due dates were updated using longest path calculations.

For our modified instances, we first spotted all release and due dates that were not obtained via longest path calculations in the original instances, and we fixed them in our modified instances. We then tightened all time windows using longest path calculations adapted to overlapping. For a WP, these calculations give the earliest start date (respectively latest finish date) knowing the earliest start dates (respectively latest finish dates) of all its predecessors (respectively successors) and the maximum possible overlapping with each predecessor (respectively successor). With our adapted calculations of release and due dates, we obtain the same time windows as the original

instances when fixing $\omega = 0$ (no overlapping), and possibly larger time windows when $\omega > 0$.

Note that modifying time windows increases the average slack s values of the instances of De Boer (1998). We calculated the new average slack values for the highest ω value of our new instances (worst case scenario), and found out that the difference between the new and the original values is equal to 2.35 on average, with a 99 % confidence interval of [2.18; 2.52]. This means that even in the worst case, new instances coming from the same original s value are very likely to have new slack values between $s + 2.18$ and $s + 2.52$. Therefore, the original instances with average slack values of 2, 5, 10, 15, and 20 are very likely to lead to new instances with buckets of slack values in the intervals [4.18; 4.52], [7.18; 7.52], [12.18; 12.52], [17.18; 17.52], and [22.18; 22.52] (respectively). Hence our new instances can still be grouped according to the average slack value of their original instances. For the sake of clarity, we will keep using the original slack values as parameters for our modified instances.

6 Illustrative example

In this section, only two simple instances (derived from rccp192) are presented for illustration purposes. For each instance, the project consists of 10 WPs, 3 resources, and a time horizon of 23 weeks. The first instance was generated while fixing $\omega = 0$, and the second one was created with $\omega = 0.2$. Figure 4 shows the precedence relations between WPs for the two instances. Each vertex represents a WP, and each directed edge represents a predecessor–successor relationship between two WPs. When a predecessor–successor couple has two or more overlapping modes, a label appears on the edge showing the number of overlapping modes between the two WPs.

In the case where $\omega = 0$, no overlapping is allowed between any couple. Thus, all of the 12 predecessor–successor couples have one overlapping mode (no overlapping). When $\omega = 0.2$, the instance generator chooses $\lceil 0.2 \cdot 12 \rceil = 3$ couples and randomly gives 2 or 3 overlapping modes for each chosen couple. As illustrated in Fig. 4, couples WPs 1–4 and WPs 6–10 have 2 overlapping modes each: WPs 4 and 10 can both start after 80, or 100 % of WPs 1 and 6 (respectively) are completed. Couple WPs 2–5 have 3 overlapping modes: WP 5 can start after 60, 80, or 100 % of WP 2 is attained. Thus WPs 1 and 6 have 2 overlapping modes with their successors, while WP 2 has 3 overlapping modes. Furthermore, each of WPs 1, 2 and 6 is divided into 2 parts, as they only have one successor that can overlap.

We implemented our model using the Optimization Programming Language (OPL) and ran our model on IBM ILOG CPLEX Optimization Studio 12.5.1.0 while minimizing a tradeoff between project cost and makespan.

Figure 5 shows the charts that we obtained with both instances, as well as the allocations for the three resources. WPs are shown in descending order in the Gantt chart (WP 1 is the highest, and WP 10 is the lowest). For each WP, a white rectangle delimits the allowed time window. Note that these rectangles are wider for $\omega = 0.2$ because release and due dates were recalculated in order to take into account the possibility of overlapping. Colored rectangles show starting and ending dates of WPs (or possibly of parts of WPs). Resource allocations are shown per period with stacked

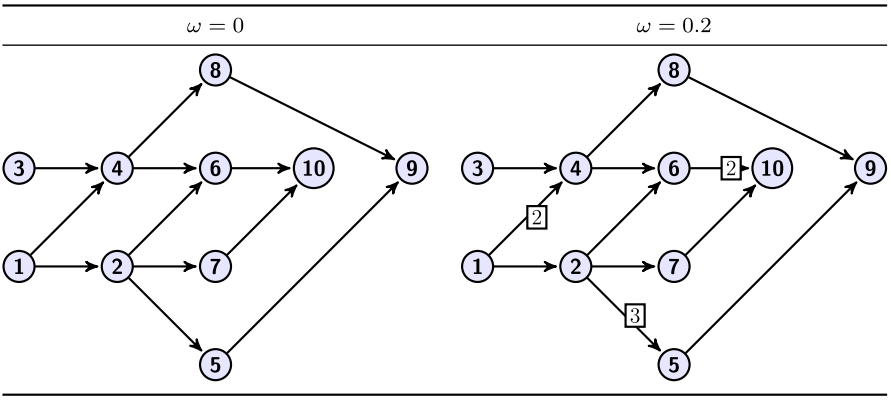

Fig. 4 Precedence relations between WPs for a project with and without overlapping

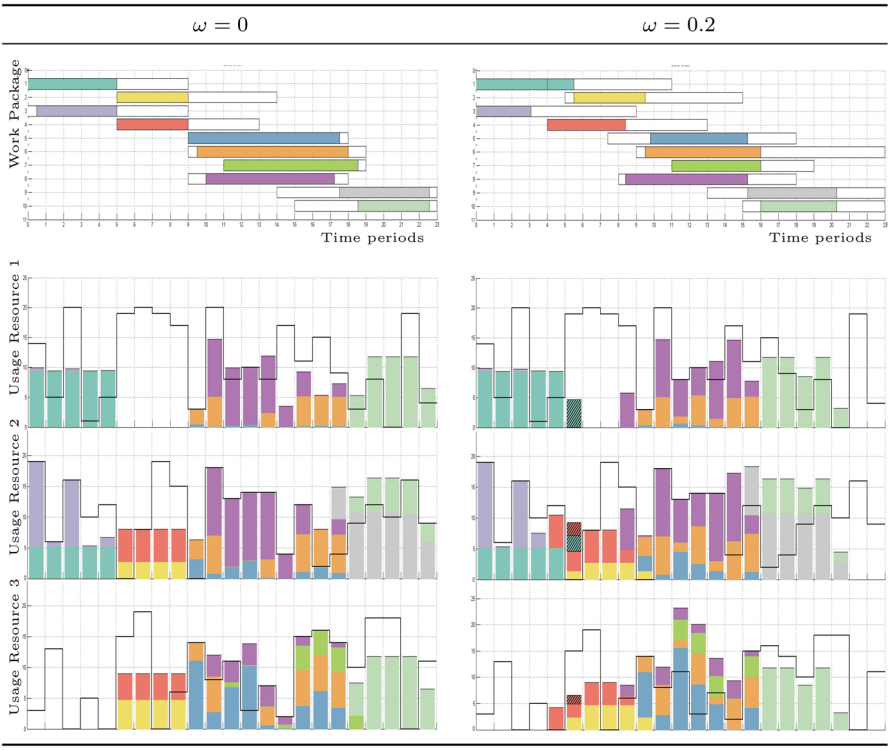

Fig. 5 Comparison of the plan obtained for a project with and without overlapping: Gantt charts (*top charts*), and stacked bar charts (*bottom three charts*) showing the usage per period of the three resources

bar charts, where each color relates to a WP in the Gantt chart, and where hatched bars refer to rework. Stepped curves represent regular resource availability for each period.

Both instances were solved to optimality in less than 4 s. When allowing overlapping, project makespan was reduced at the price of increasing project cost. In Fig. 5, we

see that WP 1 overlaps on WP 4 thus entailing rework workloads for both WPs. This additional workload is allocated during period 6. Note that, because of overlapping, WPs 8 and 9 finished earlier, thus reducing project duration by 2.3 periods (11 %). Meanwhile, project cost is increased by 43.3 units (34 %) because of a greater use of non regular resources.

However, minimizing tradeoff function does not always have the same effect on project cost and makespan as in this example. Other instances showed that overlapping can reduce the use of non regular resources, thus diminishing the project cost, at the expense of an increased duration.

7 Computational results

We performed all of our tests on a single thread, using a computational grid consisting of 26 PCs with two 3.07 GHz Intel(R) Xeon(R) X5675 Processors running under Linux. We encoded our model using the Optimization Programming Language (OPL) and ran our model on IBM ILOG CPLEX Optimization Studio 12.5.1.0 using the default values for all CPLEX parameters.

7.1 Tests on the original instances

A series of tests was conducted in order to evaluate the performance of our new model. We denote with A the original model of Haït and Baydoun (2012) and B our new model that handles overlapping. We ran the model A on the 450 instances of De Boer (1998), and our model B on our modified instances of De Boer (1998) where we fixed $\omega = 0$. In this particular case where no overlapping couples are allowed, the two sets of instances are equivalent. The objective is to minimize the cost. Having the same optimal solutions, we only compared the performance of the two models in terms of CPU times, the gaps between lower and upper bounds, and the number of times the models were solved to optimality. Table 5 shows for each class the average CPU time for the original model A and our new model B, with the objective of minimizing project cost, while Table 6 compares the number of times both models A and B found an optimal solution for each class.

Tables 5 and 6 show that the performance of the new model B is slightly degraded compared to model A when tested on equivalent instances. This may be due to the fact that model A is more compact and adapted to the case without overlapping. Nevertheless, these instances are difficult to solve and our results can be compared to those of Hans (2001) (branch and price for the RCCP) and Kis (2005) (branch and cut for the RCPSVP), as presented in Haït and Baydoun (2012).

7.2 Tests on the modified instances

We ran our new model B on five sets, with five different values of ω (0, 0.1, 0.2, 0.3 and 0.4), of 450 instances each (total of 2250 instances), with a time limit of 10,000 s for every test. We then divided the time limit into 20 intervals of 500 s each, and counted

Table 5 Average CPU time in seconds for models *A* and *B* when minimizing the project cost without overlapping

	$n = 10$			$n = 20$			$n = 50$		
$r =$	3	10	20	3	10	20	3	10	20
$s = 2$									
A	0.1	0.1	0.1	0.1	0.2	0.2	0.4	0.6	1
B	0.1	0.1	0.1	0.2	0.2	0.2	0.7	0.9	1.1
$s = 5$									
A	0.3	0.4	0.4	1	2.1	3.8	14	289	280
B	0.3	0.4	0.5	1.6	2.5	4	30	406	346
$s = 10$									
A	3.5	2.5	3	39	223	235	2694	4930	5000
B	4.9	2.7	3.1	56	383	187	3488	5000	5000
$s = 15$									
A	6	10	22	450	2147	3115	5000	5000	5000
B	11	12	21	636	2217	2980	5000	5000	5000
$s = 20$									
A	13	66	48	1036	4198	3818	5000	5000	5000
B	22	74	48	2071	4315	3771	5000	5000	5000

Table 6 Number of times models *A* and *B* were solved to optimality when minimizing the project cost without overlapping

	$n = 10$			$n = 20$			$n = 50$		
$r =$	3	10	20	3	10	20	3	10	20
$s = 2$									
A	10	10	10	10	10	10	10	10	10
B	10	10	10	10	10	10	10	10	10
$s = 5$									
A	10	10	10	10	10	10	10	10	10
B	10	10	10	10	10	10	10	10	10
$s = 10$									
A	10	10	10	10	10	10	6	1	0
B	10	10	10	10	10	10	4	0	0
$s = 15$									
A	10	10	10	10	7	7	0	0	0
B	10	10	10	10	8	6	0	0	0
$s = 20$									
A	10	10	10	10	4	3	0	0	0
B	10	10	10	8	2	4	0	0	0

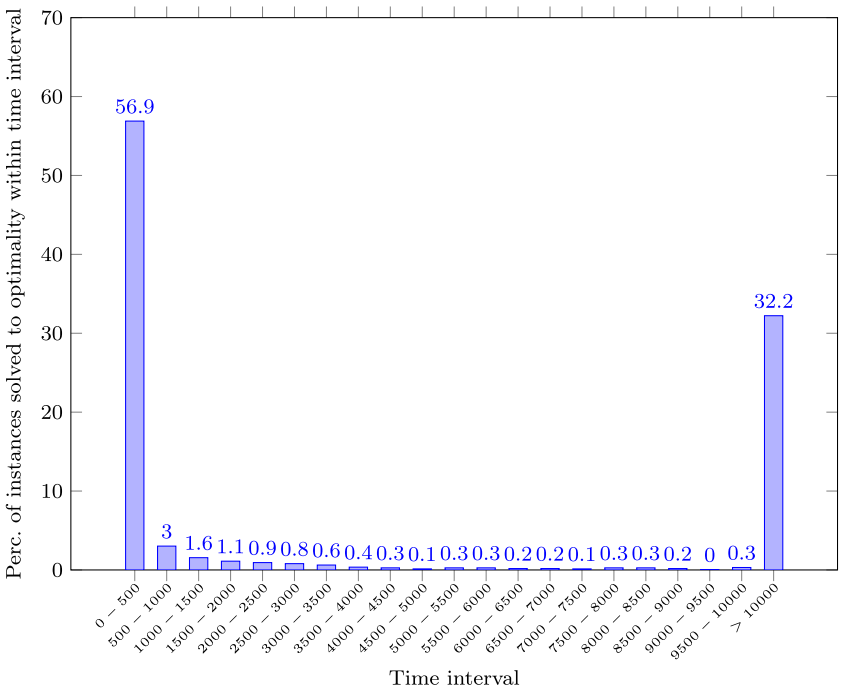

Fig. 6 Percentage of instances that were solved to optimality within the specified time interval

the number of times CPU time was inside each interval. Figure 6 shows that 56.9 % of instances were solved to optimality during the first 500 s, and that time limit was reached before optimality for 32.2 % of instances. For 0.2 % of instances, no feasible (integer) solution was found after 10,000 s of computation. Figure 6 also shows that only 2 % of instances were solved to optimality between 5000 and 10,000 s. Consequently, in the remaining tests, we terminated the search after 5000 s for every test.

We define five criteria for evaluating the performance of the model for each value of ω: the percentage of instances that were not solved to optimality, the average CPU time divided by time limit, the percentage of instances that needed more than 50 % of the time limit in order to be solved to optimality, the average gap, and the percentage of instances that have more than a 5 % gap when reaching the time limit. Figure 7 presents the results for each value of ω in a radar chart, where better performing set of instances are closer to the center. This chart contains five spokes representing the five criteria. Each colored line represents the results for one value of ω. Continuous lines give the values after 5000 s of calculation time, while dashed lines give the results after 10,000 s of calculation time. Figure 7 shows that the performance of our model is degraded on average, according to the five defined criteria, when the percentage of predecessor–successor couples that can overlap augments. This is the result of an increase in the number of binary and continuous variables, due to the separation of some WPs into several parts, and also due to the different possible overlapping modes.

It is clear that the new parameter ω is not the only factor that has an impact on CPU time; all four of the aforementioned parameters of instances can affect the performance

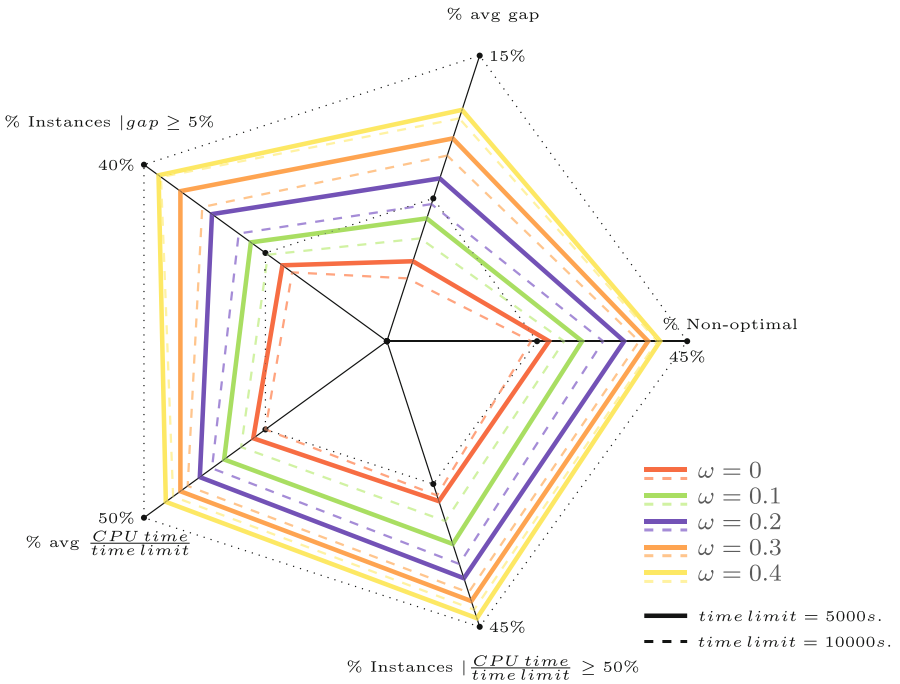

Fig. 7 Impact of changing ω on the performance of B, when minimizing project cost

of the model. Appendix 1 provides an extended statistical study of the influence on CPU time of the parameters.

7.3 Comparison of objective functions

In order to conduct an analysis on the results that were obtained, we ran our model on the 450 instances with $\omega = 0$ (equivalent to the original instances of De Boer 1998) two times. We first fixed the objective of minimizing project cost, and then we ran the model another time while minimizing project delivery time. Among 450, only 340 were solved to optimality when minimizing project cost. For these instances, we obtained the optimal cost Cost_0 and makespan Makespan_0. We integrated these two parameters in all of the corresponding data files (with the five different values of ω). This means that for a given instance, we added two additional parameters, Cost_0 and Makespan_0, which are the minimum possible cost and makespan for the instance if no overlapping was allowed.

We used these two parameters in order to create a tradeoff objective function (54) with normalized project cost and delivery time.

$$\text{minimize} \quad \beta_{\text{cost}} \cdot \frac{\text{cost}}{\text{Cost}_0} + \beta_{\text{makespan}} \cdot \frac{\text{makespan}}{\text{Makespan}_0} \tag{54}$$

Table 7 Three combinations for coefficients β_{cost} and β_{makespan}

	β_{cost}	β_{makespan}
Combination 1	25	75
Combination 2	50	50
Combination 3	75	25

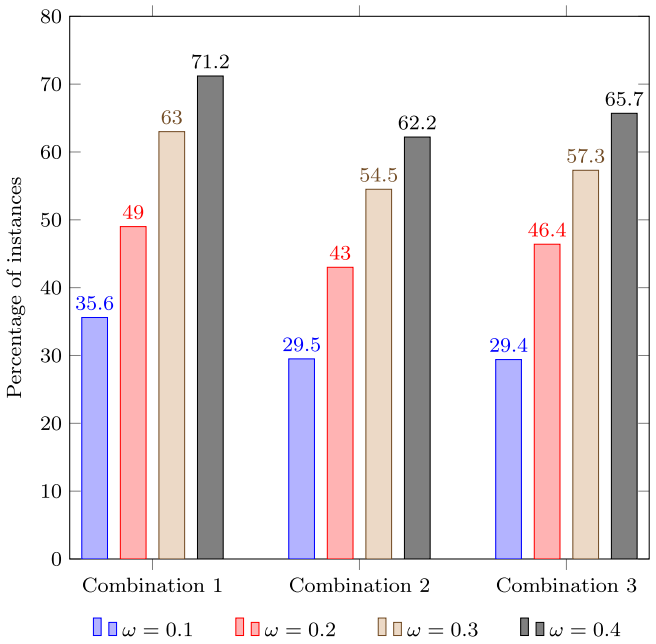

Fig. 8 Percentage of instances where overlapping improved the objective

For each instance we tested three combinations for the coefficients β_{cost} and β_{makespan} as depicted in Table 7. For each combination, and each non-zero value of ω, we compared the results to the case in which no overlapping was allowed ($\omega = 0$). Figure 8 shows for each combination and non-zero value of ω, the percentage of instances where the objective was improved compared to the case where no overlapping was allowed.

Two remarks can be made concerning the shape of the bar graph 8. First, note that, for each combination of values for β_{cost} and β_{makespan}, the percentage of instances where the objective was improved when overlapping was allowed increases with ω. This can be explained by the fact that the bigger ω is, the more flexible the problem becomes. Thus, it is more likely to have a smaller objective value with a bigger ω. Secondly, note that for a given ω, the percentage of instances that have a better objective with overlapping is bigger when $\beta_{\text{cost}} = 25$ and $\beta_{\text{makespan}} = 75$ (combination 1) than the case in which $\beta_{\text{cost}} = 75$ and $\beta_{\text{makespan}} = 25$ (combination 3). This can be explained by the fact that overlapping is mainly beneficial when accelerating project delivery time. Thus, its favorable effects are more pronounced when the tradeoff puts

more emphasis on makespan than on cost. However, even though overlapping entails additional workload, it is still beneficial for having a smaller project cost, by adding a flexibility in distributing the workload during regular resource capacities.

8 Conclusion

Motivated by the common use of concurrent engineering methods in construction projects, we proposed an interesting extension of the RCCP that allows for WPs to overlap. Five sets of 450 modified RCCP instances were created and let us conduct two types of analysis on our model. The performance analysis showed that our model is concurrent with the original model when our modified instances are equivalent to the original ones, and becomes less efficient when the percentage of possible overlapping couples increases. However, this slower performance is the price to pay in order to have smaller project delivery time and cost, as it was shown by our results analysis. In fact, our results showed that overlapping is beneficial for accelerating project delivery times, and also for reducing project cost by allowing a better distribution of workload.

Our model could be used as a tool of great interest to practitioners during tactical planning of projects. Existing tools do not consider overlapping, forcing planners to rely on their experience and intuition in order to decide how WPs should overlap in a project. The proposed solution allows them to make decisions in a rigorous manner. Moreover, the model allows planning with several possible targets. Thus, it is possible to quickly generate multiple project plans with different objectives and let managers choose the plan that best suits them. Finally, within a framework of hierarchical planning, the model allows planners to determine start and end dates of WPs and the required amounts of resources per period. Information found at a tactical level would then be used as constraints when scheduling the activities at an operational level (Kis and Kovács 2012).

Future work could focus on improving the performance of the model by exploring alternative modeling and solving techniques. Another avenue for research could be the development of a model-based heuristic dedicated for the hard instances, in order to provide acceptable solutions in a reasonable time.

Acknowledgments This work has been supported by the Natural Sciences and Engineering Research Council of Canada and the Jarislowsky/SNC-Lavalin Research Chair in the Management of International Projects. Their support is gratefully acknowledged.

Appendix

Statistical analysis of the influence of the parameters on CPU time

This section provides an analysis of the influence of parameters s, r, n, and ω on CPU times for the tests with the modified instances presented Sect. 7.2. Figure 9 shows for each parameter the average CPU time, with a 95 % confidence interval. It suggests that parameters n and s have the biggest impact on the performance, followed by ω,

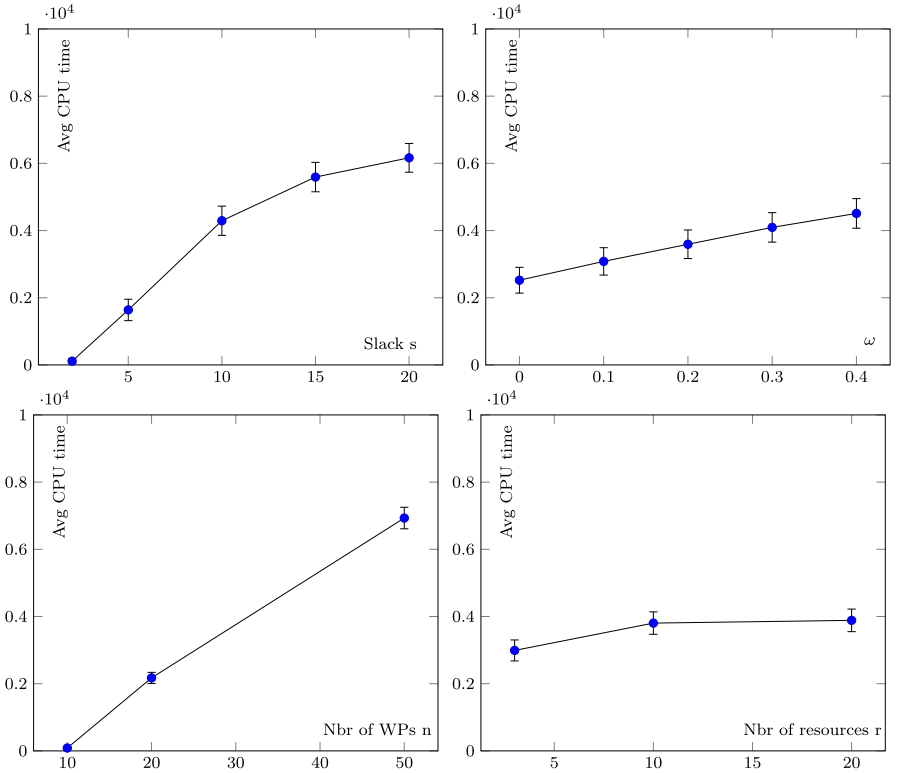

Fig. 9 Average CPU time, with 95 % confidence level for different values of four instance parameters

and lastly r. The following sections explain our statistical analysis of the effect of the four parameters s, r, n, and ω on CPU time.

Statistical modeling of CPU

The complete data set is composed of 2250 observations of CPU time (in seconds) with 225 different combinations of s, n, ω, and r repeated 10 times each. The 2250 cases are split into 2 sets: 725 cases where CPU time reaches the time limit of 10,000 s (when the calculations were terminated), and 1525 cases where CPU time is below 10,000. The overall distribution of the 2250 cases is very bimodal as can be seen in Fig. 6.

This highly bimodal distribution is a challenge when trying to model the statistical behavior of CPU with respect to the 4 input parameters s, n, ω, and r. Our strategy is to model the behavior of CPU against the input parameters for 3 sets:

Set 1: All 2250 observations, replacing CPU with 2 values (0 for CPU less than 10,000, and 1 for CPU equal to 10,000), and using a logistic regression model in Sect. 1.

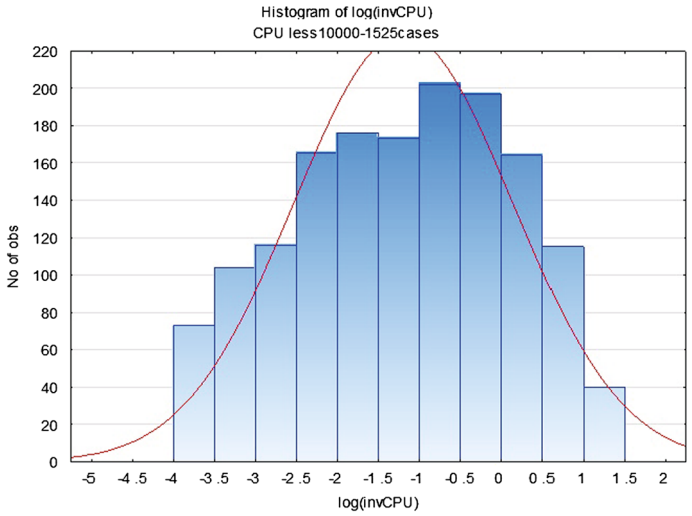

Fig. 10 Distribution of the new response variable $\log(\frac{1}{\text{CPU}})$

Set 2: Observations with CPU equal 10 000, analyzed with interaction graphical plots in Sect. 1.
Set 3: The 1525 observations with CPU less than 10,000. CPU is transformed to $\log(\frac{1}{\text{CPU}})$ and a second degree polynomial regression model is used to analyze the data in Sect. 1. This transformation was chosen after exploring several alternatives, and it was deemed the best transformation giving a bell-shaped distribution. Moreover, the chosen transformation gave the greatest explanatory model with a multiple regression R^2 equal to 0.85, greater than all other transformations. Figure 10 shows the distribution of the new response variable $\log(\frac{1}{\text{CPU}})$.

Logistic modeling of CPU with all the observations

In order to understand the impact of the 4 input parameters s, n, ω, and r on reaching the time limit, we used a logistic regression model by replacing the CPU response variable with a new variable called CPUover. This variable takes the value 0 when CPU is less than 10,000 and 1 when CPU is equal to 10,000. Our aim is to identify the input parameters that are mostly responsible for not being able to solve an instance within the time limit of 10,000. The results of the logistic regression model are given in Table 8.

The importance of a factor is directly proportional to the model coefficient and to the Odd Ratio (defined as the exponential of the regression coefficients). According to Table 8, the two most important factors are s and n, meaning that they have the biggest impact on the response CPU. The other two factors ω and r have comparable importance, and are very much less important than s and n. One way of measuring the performance of the logistic model is with the Receiver Operating Characteristic (ROC) curve. This curve plots the sensitivity and the specificity of the model (see

Table 8 Coefficients of the logistic first order model

Independent variable	Regression coefficient $b(i)$	Standard error $Sb(i)$	Lower 95 % confidence limit	Upper 95 % confidence limit	Odds ratio $\exp(b(i))$
B0: Intercept	−15.27113	0.77801	−16.79601	−13.74626	0.00000
B1: n	0.20407	0.01079	0.18294	0.22521	1.22639
B2: ω	0.07244	0.00650	0.05969	0.08518	1.07513
B3: r	0.07661	0.01194	0.05321	0.10002	1.07962
B4: s	0.49817	0.02720	0.44485	0.55149	1.64571

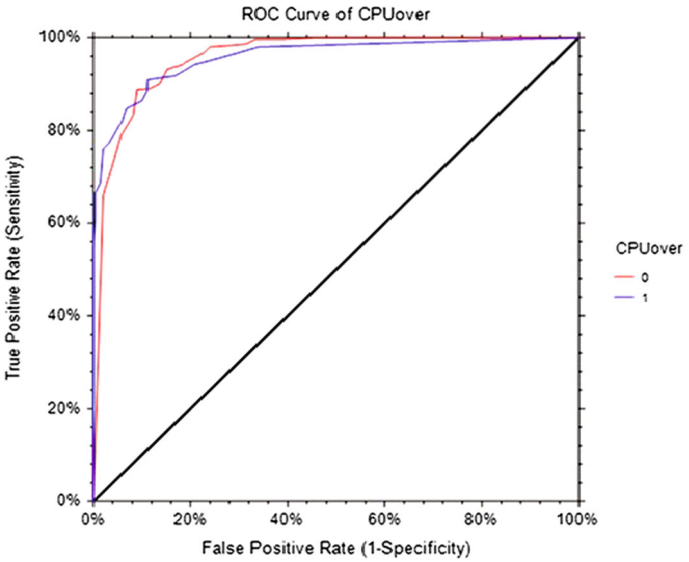

Fig. 11 Receiver operating characteristic (ROC) curve

Fig. 11). The area under the ROC Curve is equal to 0.96, which indicates a very good prediction model for discriminating between the two sets of complementary values of CPUover.

A graphical representation of observations with CPU equal to 10,000 s

The dataset is composed of 725 instances where the CPU reached the time limit of 10,000 s. None of these cases involved $n = 10$, irrespective of the values of s, ω, and r. The best way to understand the influence of the factors to reach the time limit is with an interaction graph of the different combinations of s, ω, r, and n. Two graphs are given: Fig. 12 for $n = 20$ and Fig. 13 for $n = 50$.

The plots show that the factors n and s are the major contributing factors for CPU to reach the time limit, while factors r and ω are playing a less important role.

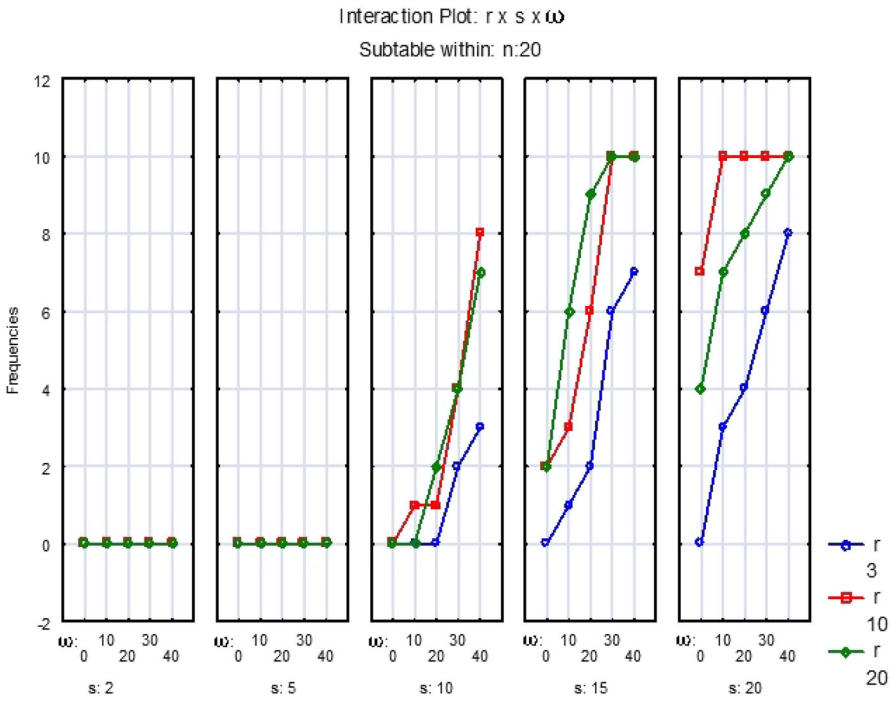

Fig. 12 Interaction plot of $r \times s \times \omega$ with $n = 20$

A regression model for cases with CPU less than 10,000 s

As mentioned before, our approach is to use a second degree polynomial regression model to analyze the transformed variable $\log(\frac{1}{\text{CPU}})$. The regression model is composed of 4 main effects (s, ω, r, n) 4 quadratic effects $(ss, \omega\omega, rr, nn)$, and 6 interaction effects $(\omega r, \omega s, \omega n, rs, rn, sn)$. Table 9 gives the results of their parameter estimates in coded forms (Effect) and in their natural units (Regressn Coeff.). The relative importance of the effects is given in the Pareto Graph (Fig. 14) and is based on the absolute values of the t statistics given in Table 9.

The interpretation of the relative importance of the factors is very clear: s and n are the driving factors, and ω comes in third place. This is followed by the quadratic effects of s and n and the interaction of s and n. The remaining factor r is significant but the analysis of variance table (not shown here) indicates that r is responsible for only 2 % of the explained variation of CPU.

In summary, all three analysis have identified that the main factors in driving the CPU time are s, n and ω in that order, and that factor r is not very important. Of course are conclusions apply to the chosen experimental space.

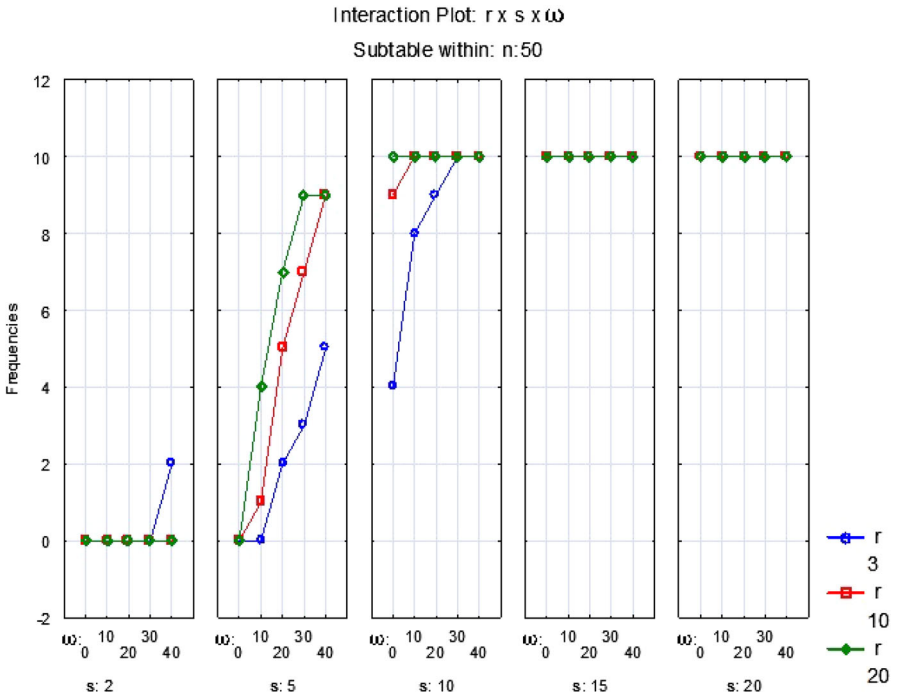

Fig. 13 Interaction plot of $r \times s \times \omega$ with $n = 50$

Table 9 Estimation of the model parameters

Factor name	Effect	Std. err.	$t(1510)$	p	Regressn coeff.
Mean/interc.	−3.86	0.052	−74.41	0.0000	3.1178
ω	−0.78	0.028	−27.65	0.0000	−0.0264
$\omega\omega$	0.01	0.016	0.81	0.4174	0.0001
s	−3.00	0.067	−44.94	0.0000	−0.2752
ss	0.46	0.023	20.04	0.0000	0.0092
n	−4.00	0.104	−38.57	0.0000	−0.1013
nn	1.20	0.088	13.66	0.0000	0.0015
r	−0.53	0.046	−11.44	0.0000	−0.0438
rr	0.32	0.060	5.38	0.0000	0.0022
ωs	0.02	0.017	1.26	0.2086	0.0002
ωn	−0.22	0.031	−7.20	0.0000	−0.0006
ωr	−0.01	0.023	−0.40	0.6898	−0.0001
sn	−1.34	0.077	−17.34	0.0000	−0.0067
sr	−0.08	0.028	−2.80	0.0051	−0.0009
nr	−0.32	0.052	−6.05	0.0000	−0.0009

Effect estimates var.: $\log(\frac{1}{\text{CPU}})$ R-sq = 0.84764 Adj: 0.84622 CPU less 10,000 − 1525 cases 4 factors, 1 blocks, 1525 runs MS Residual = 0.2715

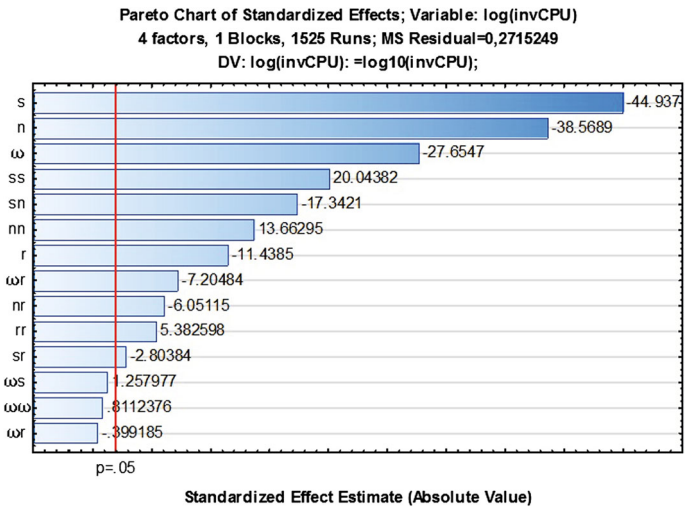

Fig. 14 Pareto plot of the effect importance ranking: significant effects at the red line level or above

References

Alfieri A, Tolio T, Urgo M (2011) A project scheduling approach to production planning with feeding precedence relations. Int J Prod Res 49(4):995–1020

Berthaut F, Robert P, Nathalie P, Adnène H (2014) Time-cost trade-offs in resource-constraint project scheduling problems with overlapping modes. Int J Proj Organ Manag 6(3):215–236

Bianco L, Caramia M (2013) A new formulation for the project scheduling problem under limited resources. Flex Serv Manuf J 25(1–2):6–24. doi:10.1007/s10696-011-9127-y

Cherkaoui K, Pellerin R, Baptiste P, Perrier N (2013) Planification hiérarchique de projets EPCM. In: 10ème Conférence Internationale de Génie Industriel 2013, La Rochelle, France

De Boer R (1998) Resource-constrained multi-project management: a hierarchical decision support system. PhD thesis, University of Twente, Enschede, The Netherlands

Gademann N, Schutten M (2005) Linear-programming-based heuristics for project capacity planning. IIE Trans 37(2):153–165

Gerk JEV, Qassim RY (2008) Project acceleration via activity crashing, overlapping, and substitution. IEEE Trans Eng Manag 55(4):590–601

Grèze L, Pellerin R, Leclaire P, Perrier N (2012) A heuristic method for resource-constraint project scheduling with activity overlapping. J Intell Manuf (forthcoming)

Hans E (2001) Resource loading by branch-and-price techniques. PhD thesis, University of Twente, Enschede, The Netherlands

Haït A, Baydoun G (2012) A new event-based MILP model for the resource-constrained project scheduling problem with variable intensity activities. In: The IEEE international conference on industrial engineering and engineering management 2012, Honk Kong

Kis T (2005) A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. Math Program 103(3):515–539

Kis T (2006) RCPS with variable intensity activities and feeding precedence constraints. In: Józefowska J, Weglarz J (eds) Perspectives in modern project scheduling, chap 5. Springer, Berlin, pp 105–129

Kis T, Kovács A (2012) A cutting plane approach for integrated planning and scheduling. Computers Oper Res 39(2):320–327. doi:10.1016/j.cor.2011.04.006. http://www.sciencedirect.com/science/article/pii/S0305054811001079

Kolisch R, Sprecher A, Drexl A (1995) Characterization and generation of a general class of resource-constrained project scheduling problems. Manag Sci 41(10):1693–1703

Naber A, Kolisch R (2014) MIP models for resource-constrained project scheduling with flexible resource profiles. Eur J Oper Res 239(2):335–348. doi:10.1016/j.ejor.2014.05.036

Pritsker A, Watters L (1968) A zero-one programming approach to scheduling with limited resources. Memorandum (Rand Corporation) RM-5561-PR. Rand Corporation

Wullink G (2005) Resource loading under uncertainty. PhD thesis, University of Twente, Enschede, The Netherlands