OR Spectrum (2016) 38:405–425
DOI 10.1007/s00291-016-0435-1

REGULAR ARTICLE

# Resource levelling in project scheduling with generalized precedence relationships and variable execution intensities

**Lucio Bianco**[1] · **Massimiliano Caramia**[1] · **Stefano Giordani**[1]

Received: 18 December 2014 / Accepted: 9 February 2016 / Published online: 18 February 2016


**Abstract** We study the problem of levelling resources in a project with generalized precedence relationships, given a deadline for the completion of all the activities and variable execution intensities and flexible durations of the activities. Variable execution intensities have been taken into account firstly by Kis (Math Program 103(3):515–539, 2005) applied to a real world scenario in which, due to the physical characteristics of some manufacturing processes, the effort associated with a certain activity for its execution may vary over time. Generalized precedence relationships and variable intensity execution and duration have not been dealt with together to the best of our knowledge. For this novel problem we propose a mixed-integer linear programming formulation, a lower bound based on Lagrangian relaxation, and a branch and bound algorithm. Computational results on known benchmarks are provided.

**Keywords** Project scheduling · Generalized precedence relationships · Resource levelling

✉ Massimiliano Caramia
caramia@dii.uniroma2.it

Lucio Bianco
bianco@dii.uniroma2.it

Stefano Giordani
giordani@dii.uniroma2.it

[1] Dipartimento di Ingegneria dell'Impresa, University of Rome "Tor Vergata", Via del Politecnico, 1, 00133 Rome, Italy

## 1 Introduction

Resource levelling problems play a key role in project management when one aims to keep under control the peak of resource usage over time, i.e., avoiding undesired resource overloads. From the complexity viewpoint, resource levelling problems are NP-hard in the strong sense (Neumann et al. 2003). Albeit a plethora of resource levelling functions may be defined to attain the specified goal, three objective functions are well established in the literature to cope with resource levelling. These functions, in turn, produce three resource leveling problems (Rieck and Zimmerman 2015): the *classical resource levelling problem*, where the total squared resource utilization cost has to be minimized Burgess and Killebrew (1962); the *total overload cost problem*, where costs are generated when a threshold for the resource utilization (or a given resource supply) is exceeded (Easa 1989); the *total adjustment cost problem*, where one is concerned with the minimization of the cumulative costs arising from increasing or decreasing the utilizations of resources (Kreter et al. 2014). In this paper, we study the total overload cost problem. In particular, we propose a generalization of the problem in which (i) variable intensities in the execution of the activities and (ii) flexible activity durations are allowed.

Variable execution intensity has been taken into account firstly by Kis (2005) applied to a real world scenario in which, due to the physical characteristics of some manufacturing processes, the effort associated with a certain activity for its execution may vary over time, e.g., human resources that can be shared among a set of simultaneous activities in proportion variable over time. In this case, the amount of work per time unit devoted to each activity and, consequently, also its duration are not univocally defined. More recently, Fündeling and Trautmann (2010) and Bianco and Caramia (2011) considered variable execution intensities in project scheduling. The former authors introduced the latter feature in a project scheduling problem where the aim is to determine a feasible resource-usage profile for each activity such that the project duration is minimized subject to precedence and resource-capacity constraints. They propose a priority-rule scheduling method that iteratively determines a feasible resource-usage profile for each activity. The latter authors considered variable execution intensities in a project scheduling problem with resource constraints, feeding precedence relations, and minimum makespan objective.

In this study, we consider projects modeled by activity-on-nodes networks $N = (V, A; \delta)$, where $V = V^r \cup \{0, n+1\}$ is the set of nodes formed by the set $V^r$ of $n$ real activities and two dummy activities, i.e., 0 and $n+1$ with duration equal to zero, representing project beginning and completion, respectively, and corresponding to the source and sink nodes of the project network; $A$ is the set of arcs representing *generalized precedence relations* (GPRs) between pairs of activities with weights $\delta_{ij} \in \mathbb{Z}, \forall (i, j) \in A$. GPRs are temporal constraints in which the starting/finishing times of a pair $(i, j)$ of activities have to be separated by at least or at most an amount ($d_{ij}^{\min}$ and $d_{ij}^{\max}$, respectively) of time denoted as "time lag" (minimum time lag and maximum time lag, respectively). Therefore, GPRs can be classified into *start-to-start*, *start-to-finish*, *finish-to-start*, and *finish-to-finish*. Following the results by Bartusch et al. (1988) we consider the project network in a standardized form in which only start-to-start relations are represented and if activity $j$ cannot be started earlier than

$d_{ij}^{\min}$ time units after the start time of activity $i$ then we have an arc $(i, j) \in A$ with weight $\delta_{ij} = d_{ij}^{\min}$ in $N$. If activity $j$ has to be started no later than $d_{ij}^{\max}$ time units after the start time of activity $i$ then we have an arc $(j, i) \in A$ with weight $\delta_{ji} = -d_{ij}^{\max}$. Arc set $A$ includes arcs $(0, i)$ between dummy node 0 and each node $i \in V^r$, with weight $\delta_{0i} = 0$, and arcs $(i, n+1)$ between each node $i \in V^r$ and dummy node $n+1$, with weight $\delta_{i,n+1}$ equal to the duration of activity $i$. Besides GPRs, a deadline $\bar{d}$ within which all the activities have to be completed is given: we assume that network $N$ contains arc $(n+1, 0)$ with weight $\delta_{n+1,0} = -\bar{d}$ to force this requirement.

Therefore, the problem tackled in our work is as follows: given a set of $K$ renewable resources, with $Y_k$ and $c_k$ being, respectively, the available amount of resource type $k$ and its additional resource unitary cost per time period, a set of $n$ activities, with activity $i$ requiring a total amount $\bar{r}_{ik}$ of resource $k$ and having a duration ranging between a minimum and a maximum value $p_i^{\min}$ and $p_i^{\max}$, respectively, and a set of GPRs constraints, we want find a schedule, where activities are processed with variable execution intensity, minimizing the total additional resource cost within the deadline $\bar{d}$. To the best of our knowledge this is a novel problem.

Reviewing the literature of the total overload cost problem (without the generalization proposed in this paper), we can observe that the first exact approach for resource levelling with precedence constraints is due to Petrovic (1969). The method is based on the enumeration of the feasible integral start times of project activities and is tailored for the classical resource levelling problem. Engelhardt and Zimmermann (1998), exploiting an idea of Ahuja (1976), proposed a method that enumerates all combinations of activity start times to minimize the sum over time of the squared changes in the resource utilization and several other objective functions. Nübel (2001) presented a tree-based enumeration approach for the resource renting problem, and Neumann et al. (2003) outlined how this approach can be used to solve resource levelling problems. Gather and Zimmermann (2009) have sketched some weaknesses of the latter approach and developed a new and more efficient procedure, relying on the paper of Gabow and Myers (1978). The algorithms proposed by Rieck et al. (2012) are the best known for the classical resource levelling and total overload cost problems, while those proposed by Kreter et al. (2014) are the best known for the total adjustment cost problem. Additional literature on resource levelling problems with GPRs can be found in Schwindt and Zimmerman (2015).

The paper is organized as follows. In Sect. 2, we propose a mathematical formulation for the problem under consideration. In Sect. 3, we calculate a Lagrangian-based lower bound based on the latter formulation. In Sect. 4, we describe the branch and bound scheme proposed to solve the mathematical formulation, and, finally, in Sect. 5, we show computational results on known benchmarks.

## 2 The mathematical model

In the following, we assume that the planning horizon is discretized into $\bar{d}$ unit-width time periods $[0, 1), [1, 2), \ldots, [\bar{d}-1, \bar{d})$, indexed by $t = 1, \ldots, \bar{d}$, respectively. Let us define the following parameters:

- $K$ the number of renewable (continuously divisible) resources;

- $Y_k$ the number of available units of resource of type $k$, with $k = 1, \ldots, K$;
- $c_k$ the unitary cost of additional resources of type $k$ per time period;
- $\bar{r}_{ik}$ the overall amount of units of resource $k$ necessary to carry out activity $i$;
- $p_i^{\max}$ the maximum duration of activity $i$;
- $p_i^{\min}$ the minimum duration of activity $i$, with $0 < p_i^{\min} \leq p_i^{\max}$;
- $A$ the set of ordered pairs of activities with temporal relations;
- $\delta_{ij}$ the lag between $(i, j) \in A$;
- $\bar{d}$ the deadline of the project.

Furthermore, let us consider the following decision variables:

- $u_{kt}$ the type $k$ resource usage during time period $t$;
- $x_{it}$ the fraction of activity $i$ executed within the end of time period $t$;
- $s_{it}$ a binary variable that assumes value 1 if activity $i$ starts at the beginning of time period $t$ or before, and assumes value 0 otherwise;
- $f_{it}$ a binary variable that assumes value 1 if activity $i$ has finished within the end of time period $t$, and assumes value 0 otherwise.

The proposed mathematical model $\mathcal{F}$ is as follows:

$$\min \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} c_k \max(0, u_{kt} - Y_k) \tag{1}$$

$$\text{s.t. } x_{it} - x_{i,t-1} \geq \frac{1}{p_i^{\max}}(s_{it} - f_{i,t-1}), \quad i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d} \tag{2}$$

$$x_{it} - x_{i,t-1} \leq s_{it} - f_{i,t-1}, \quad i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d} \tag{3}$$

$$\sum_{t=1}^{\bar{d}} s_{it} \geq \sum_{t=1}^{\bar{d}} s_{jt} + \delta_{ij}, \quad \forall (i, j) \in A \tag{4}$$

$$s_{it} \leq s_{i,t+1}, \quad i = 0, \ldots, n+1; \quad t = 1, \ldots, \bar{d} \tag{5}$$

$$f_{i,t-1} \leq f_{it}, \quad i = 0, \ldots, n+1; \quad t = 1, \ldots, \bar{d} \tag{6}$$

$$x_{i\bar{d}} = f_{i\bar{d}} = s_{i\bar{d}} = 1, \quad i = 1, \ldots, n \tag{7}$$

$$x_{i0} = f_{i0} = 0, \quad i = 1, \ldots, n \tag{8}$$

$$f_{it} \leq x_{it}, \quad i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d} \tag{9}$$

$$x_{it} \leq s_{it}, \quad i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d} \tag{10}$$

$$f_{00} = s_{01} = 1 \tag{11}$$

$$f_{n+1,\bar{d}} = s_{n+1,\bar{d}+1} = 1 \tag{12}$$

$$\sum_{i=1}^{n} \bar{r}_{ik}(x_{it} - x_{i,t-1}) = u_{kt}, \quad k = 1, \ldots, K; \quad t = 1, \ldots \bar{d} \tag{13}$$

$$p_i^{\min} \leq 1 + \sum_{t=1}^{\bar{d}} (s_{it} - f_{it}) \leq p_i^{\max}, \quad i = 1, \ldots, n \tag{14}$$

$$u_{kt} \geq 0, \quad k = 1, \ldots, K; \quad t = 1, \ldots, \bar{d} \tag{15}$$

$$x_{it} \geq 0, \quad i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d} \tag{16}$$

$$s_{it} \in \{0, 1\}, \quad i = 0, \ldots, n+1; \quad t = 1, \ldots, \bar{d}+1 \tag{17}$$

$$f_{it} \in \{0, 1\}, \quad i = 0, \ldots, n+1; \quad t = 0, \ldots, \bar{d}. \tag{18}$$

The objective function (1) minimizes the total cost of the additional resource usage. Constraints (2) and (3) regulate the minimum and maximum fraction of activity $i$ to be executed in time period $t$. Note that when an activity $i$ is not in execution at time period $t$, i.e, $s_{it} = 0$ and $f_{i,t-1} = 0$ (activity $i$ is not started within the beginning of time period $t$) or $s_{it} = 1$ and $f_{i,t-1} = 1$ (activity $i$ is finished within the end of time period $t-1$), these constraints imply that $x_{it} - x_{i,t-1} = 0$. When, instead, $i$ is in execution at time period $t$, i.e, $s_{it} = 1$ and $f_{i,t-1} = 0$, these constraints imply that $x_{it} \geq x_{i,t-1}$. Constraints (4) model start-to-start precedence constraints with time-lags $\delta_{ij}$, $(i, j) \in A$, where $\delta_{ij}$'s may depend in general also on the durations of activities $i$ and $j$, respectively, being the duration $p_i$ of activity $i$ equal to $1 + \sum_{t=1}^{\bar{d}} (s_{it} - f_{it})$. Note that, as mentioned in the introduction, we consider standardized networks, i.e., networks in which each arc models a start-to-start temporal relation; this is possible thanks to the Bartusch *et al.*'s transformations by which one can modify a general temporal relation in one of latter type. For instance, in case of a finish-to-start relation between activities $(i, j)$ with time lag $d_{ij}^{\min}$, we have $S_j \geq F_i + d_{ij}^{\min}$, being $S_j$ and $F_i$ the starting time of $j$ and the finishing time of $i$, respectively. By observing that $F_i = S_i + p_i$ we may rewrite the latter equality as $S_j \geq S_i + p_i + d_{ij}^{\min}$ which is a start-to-start relation with time lag $\delta_{ij} = p_i + d_{ij}^{\min}$. Noting that $S_i = \bar{d} - \sum_{t=1}^{\bar{d}} s_{it}$, constraints (4) follow. Constraints (5) and (6) are congruency constraints on the values assumed by variables $s_{it}$ and $f_{it}$ over time. Constraints (7) say that every activity $i$ must start and finish within the planning horizon. Constraints (8) represent the initialization conditions for variables $x_{it}$ and $f_{it}$ when $t = 0$. Constraints (9) and (10) force $f_{it}$ to be zero if $x_{it} < 1$ and $x_{it}$ to be zero if $s_{it} = 0$. Constraints (11) and (12) are initialization conditions for dummy activities 0 and $n+1$, respectively. Resource usage is represented by relations (13). Constraints (14) force the duration of each activity $i$ to be not less than $p_i^{\min}$ and not greater than $p_i^{\max}$. Constraints (15)–(18) limit the range of variability of the variables.

In order to linearize the objective function, we introduce additional variables $\bar{u}_{kt} \geq 0, k = 1, \ldots, K, t = 1, \ldots, \bar{d}$, replace (1) by $\min \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} c_k \bar{u}_{kt}$, replace constraints (13), with

$$\bar{u}_{kt} \geq \sum_{i=1}^{n} \bar{r}_{ik}(x_{it} - x_{i,t-1}) - Y_k, \quad k = 1, \ldots, K; \quad t = 1, \ldots, \bar{d}, \tag{19}$$

and replace in constraints (15) $u_{kt}$ by $\bar{u}_{kt}$.

Recall that the deadline $\bar{d}$ may be represented as a start-to-finish constraint between activities 0 and $n+1$ with maximum time lag $\bar{d}$, and, therefore, induces an arc $(n+1, 0)$ with weight $\delta_{n+1,0} = -\bar{d}$. Denoting with

- $\mathrm{ES}_i$, a lower bound on the earliest start time of activity $i$, computed as the longest path length from node 0 to node $i$ in the project network, using $p_i^{\min}$ and/or $p_i^{\max}$ as

activity durations in order to underestimate the values of weights $\delta_{ij}$ (note that $\delta_{ij}$ may increase, decrease, or remain unchanged with increasing value of $p_i$ and/or $p_j$, according to the specific type of GPR between activities $i$ and $j$),
- $\mathrm{LS}_i$, an upper bound on the latest start time of activity $i$, computed as $\bar{d}$ minus the longest path length from node $i$ to node $n+1$ in the project network (which equals the negative of the longest path length from node $i$ to node 0), considering likewise underestimations of $\delta_{ij}$ values,
- $\mathrm{EF}_i$, a lower bound on the earliest finish time of activity $i$, i.e., $\mathrm{ES}_i + p_i^{\min}$,
- $\mathrm{LF}_i$, an upper bound on the latest finish time of activity $i$, i.e., $\mathrm{LS}_i + p_i^{\max}$,

the ranges of variability $t = 1, \ldots, \bar{d}$ in the constraints of our model may be tightened according to latter values. Time windows will not be directly used in the definition of $\mathcal{F}$; rather, as explained in Sect. 4, they will be considered by the branch and bound algorithm to strengthen the branching phase.

Finally, we note that our model is able to take into account available amounts of resources varying with time, i.e., $Y_{kt}$ instead of $Y_k$, as well as different unitary costs $c_{kt}$ per period for the usage of additional units of resources, and that amounts $Y_k$ and costs $c_k$ are considered both for ease of presentation and for compliance with data presented in benchmarks.

## 3 A Lagrangian-based lower bound

In this section, we will show how is it possible to estimate Lagrangian multipliers $\lambda_{kt} \geq 0$, with $k = 1, \ldots, K; \quad t = 1, \ldots, \bar{d}$, in the attempt of solving the Lagrangian dual problem related to the Lagrangian relaxation ($\mathcal{LR}_{\mathcal{F}}$) of constraints (19) of the latter formulation $\mathcal{F}$. The approach used here follows the same lines of what done in Bianco and Caramia (2011, 2012).

Let us start by writing $\mathcal{LR}_{\mathcal{F}}$:

$$\omega(\lambda) = \min \left\{ \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} c_k \bar{u}_{kt} - \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} \lambda_{kt} \left[ - \sum_{i=1}^{n} \bar{r}_{ik} (x_{it} - x_{i,t-1}) + Y_k + \bar{u}_{kt} \right] \right\}$$

$$\text{s.t.} \quad x_{it} - x_{i,t-1} \geq \frac{1}{p_i^{\max}} (s_{it} - f_{i,t-1}), \quad i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d}$$

$$\ldots$$

$$\ldots$$

$$p_i^{\min} \leq 1 + \sum_{t=1}^{\bar{d}} (s_{it} - f_{it}) \leq p_i^{\max}, \quad i = 1, \ldots, n$$

$$\bar{u}_{kt} \geq 0, \quad k = 1, \ldots, K; \quad t = 1, \ldots, \bar{d}$$

$$x_{it} \geq 0, \quad i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d}$$

$$s_{it} \in \{0, 1\}, \quad i = 0, \ldots, n+1; \quad t = 1, \ldots, \bar{d}+1$$

$$f_{it} \in \{0, 1\}, \quad i = 0, \ldots, n+1; \quad t = 0, \ldots, \bar{d}.$$

Each $\lambda_{kt} \geq 0$ in the above formulation offers a lower bound to our problem. The goal is therefore to solve the Lagrangian dual problem which maximizes the lower bound $\omega(\lambda)$ as follows:

$$\max_{\lambda_{kt} \geq 0} \left\{ \min \left[ \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} c_k \bar{u}_{kt} - \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} \lambda_{kt} \left( - \sum_{i=1}^{n} \bar{r}_{ik} (x_{it} - x_{i,t-1}) + Y_k + \bar{u}_{kt} \right) \right] \right\},$$

that is,

$$\max_{\lambda_{kt} \geq 0} \left\{ - \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} Y_k \lambda_{kt} + \min \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} \left[ c_k \bar{u}_{kt} + \lambda_{kt} \left( \sum_{i=1}^{n} \bar{r}_{ik} (x_{it} - x_{i,t-1}) - \bar{u}_{kt} \right) \right] \right\},$$

that can be rewritten as

$$\max_{\lambda_{kt} \geq 0} \left\{ - \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} Y_k \lambda_{kt} + \min \left[ \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} \bar{u}_{kt} (c_k - \lambda_{kt}) + \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} \lambda_{kt} \sum_{i=1}^{n} \bar{r}_{ik} (x_{it} - x_{i,t-1}) \right] \right\}.$$

In order to avoid a trivial lower bound value, i.e., $-\infty$, since $\bar{u}_{kt} \geq 0$, we search for $\lambda_{kt}$ values such that

$$c_k - \lambda_{kt} \geq 0, \quad k = 1, \ldots, K; \quad t = 1, \ldots, \bar{d}.$$

This leads to the following lower-bound formulation

$$\max_{\lambda_{kt}} \left\{ - \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} Y_k \lambda_{kt} + \min \left[ \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} \sum_{i=1}^{n} \lambda_{kt} \bar{r}_{ik} (x_{it} - x_{i,t-1}) \right] \right\} \tag{20}$$

$$\begin{aligned}
\text{s.t.} \quad & x_{it} - x_{i,t-1} \geq \frac{1}{p_i^{\max}} (s_{it} - f_{i,t-1}), \quad i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d} \\
& \ldots \\
& \ldots \\
& p_i^{\min} \leq 1 + \sum_{t=1}^{\bar{d}} (s_{it} - f_{it}) \leq p_i^{\max}, \quad i = 1, \ldots, n \\
& x_{it} \geq 0, \quad i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d} \\
& \lambda_{kt} \leq c_k, \quad k = 1, \ldots, K; \quad t = 1, \ldots, \bar{d} \\
& \lambda_{kt} \geq 0, \quad k = 1, \ldots, K; \quad t = 1, \ldots, \bar{d} \\
& s_{it} \in \{0, 1\}, \quad i = 0, \ldots, n+1; \quad t = 1, \ldots, \bar{d} + 1 \\
& f_{it} \in \{0, 1\}, \quad i = 0, \ldots, n+1; \quad t = 0, \ldots, \bar{d}.
\end{aligned}$$

Since by constraint (2) we have that

$$x_{it} - x_{i,t-1} \geq \frac{1}{p_i^{\max}}(s_{it} - f_{i,t-1}), \quad i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d}$$

and, by constraint (6), that

$$f_{i,t-1} \leq f_{it}, \quad i = 0, \ldots, n+1; \quad t = 1, \ldots, \bar{d},$$

we can write the following:

$$x_{it} - x_{i,t-1} \geq \frac{1}{p_i^{\max}}(s_{it} - f_{i,t-1}) \geq \frac{1}{p_i^{\max}}(s_{it} - f_{it}), \; i = 1, \ldots, n; \quad t = 1, \ldots, \bar{d}.$$

Summing up over $t$ the left-hand and the right-hand sides, we obtain

$$\sum_{t=1}^{\bar{d}}(x_{it} - x_{i,t-1}) \geq \sum_{t=1}^{\bar{d}} \frac{1}{p_i^{\max}}(s_{it} - f_{it}) = \frac{1}{p_i^{\max}} \sum_{t=1}^{\bar{d}}(s_{it} - f_{it}), \quad i = 1, \ldots, n.$$

By constraint (14) it must be

$$\sum_{t=1}^{\bar{d}}(s_{it} - f_{it}) \geq p_i^{\min} - 1, \quad i = 1, \ldots, n,$$

therefore, we have

$$\sum_{t=1}^{\bar{d}}(x_{it} - x_{i,t-1}) \geq \frac{1}{p_i^{\max}}(p_i^{\min} - 1), \quad i = 1, \ldots, n,$$

that is equivalent to

$$\sum_{t=1}^{\bar{d}}(x_{it} - x_{i,t-1}) \geq \sum_{t=1}^{\bar{d}} \frac{1}{\bar{d}} \frac{(p_i^{\min} - 1)}{p_i^{\max}}, \quad i = 1, \ldots, n.$$

By multiplying by $\lambda_{kt}$, which is non-negative, each term of the summation at the left-hand and the right-hand side, we have

$$\sum_{t=1}^{\bar{d}} \lambda_{kt}(x_{it} - x_{i,t-1}) \geq \sum_{t=1}^{\bar{d}} \lambda_{kt} \frac{1}{\bar{d}} \frac{(p_i^{\min} - 1)}{p_i^{\max}}, \quad i = 1, \ldots, n; \quad k = 1, \ldots, K.$$

Similarly, since $\bar{r}_{ik} \geq 0$, we multiply both sides of the previous inequality by the latter parameter obtaining

$$\bar{r}_{ik} \sum_{t=1}^{\bar{d}} \lambda_{kt}(x_{it} - x_{i,t-1}) \geq \bar{r}_{ik} \sum_{t=1}^{\bar{d}} \lambda_{kt} \frac{1}{\bar{d}} \frac{(p_i^{\min} - 1)}{p_i^{\max}}, \quad i = 1, \ldots, n; \quad k = 1, \ldots, K.$$

Finally, we sum both sides over $i$ and $k$ which leads to

$$\sum_{i=1}^{n} \sum_{k=1}^{K} \sum_{t=1}^{\bar{d}} \bar{r}_{ik} \lambda_{kt}(x_{it} - x_{i,t-1}) \geq \sum_{i=1}^{n} \sum_{k=1}^{K} \sum_{t=1}^{\bar{d}} \frac{\bar{r}_{ik} \lambda_{kt}(p_i^{\min} - 1)}{\bar{d} \cdot p_i^{\max}}. \tag{21}$$

By minoring the objective function (20) by means of (21), we have

$$\max_{\lambda_{kt}} \left\{ -\sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} Y_k \lambda_{kt} + \min \left[ \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} \sum_{i=1}^{n} \lambda_{kt} \bar{r}_{ik}(x_{it} - x_{i,t-1}) \right] \right\}$$
$$\geq \max_{\lambda_{kt}} \left\{ -\sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} Y_k \lambda_{kt} + \sum_{t=1}^{\bar{d}} \sum_{k=1}^{K} \sum_{i=1}^{n} \frac{\bar{r}_{ik} \lambda_{kt}(p_i^{\min} - 1)}{\bar{d} \cdot p_i^{\max}} \right\},$$

that, in turn, leads to the following formulation

$$\begin{aligned} \max \quad & \left\{ \sum_{k=1}^{K} \sum_{t=1}^{\bar{d}} \lambda_{kt} \left( \sum_{i=1}^{n} \frac{\bar{r}_{ik}(p_i^{\min} - 1)}{\bar{d} \cdot p_i^{\max}} - Y_k \right) \right\} \\ \text{s.t.} \quad & \lambda_{kt} \leq c_k, \quad k = 1, \ldots, K; \quad t = 1, \ldots, \bar{d} \\ & \lambda_{kt} \geq 0, \quad k = 1, \ldots, K; \quad t = 1, \ldots, \bar{d} \end{aligned}$$

by which it is possible to estimate Lagrangian multipliers $\tilde{\lambda}$.

## 4 The branch and bound algorithm

We solve the proposed model by means of a branch and bound algorithm. The root node of the search tree is denoted with $P_0$ and is associated with the mathematical formulation $\mathcal{F}$. The lower bound used for this node is the one obtained by solving the Lagrangian relaxation $\mathcal{LR}_{\mathcal{F}}$ with multipliers $\tilde{\lambda}$. The linear relaxation of $\mathcal{F}$ is calculated at the root as well, and the maximum between the two lower bounds is taken. Both the two relaxations are computed by means of a commercial solver. The upper bound at the root is calculated by scheduling greedily activities with the largest-duration first criterion. Given a lower and an upper bound on the optimal solution value, the algorithm starts visiting the search tree. Each node of the tree, denoted with $P_{it}$, is associated with an activity $i$ to be finished in time period $t \leq \bar{d}$. The path from node $P_0$ to node $P_{it}$ in the search tree defines a partial (finish) schedule $F_{it}$

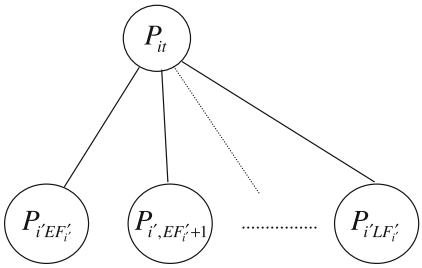


**Fig. 1** A parent node and its children

over the subset $V^r_{it} \subseteq V^r$ of scheduled activities. The $\mathrm{ES}_i$ and $\mathrm{LS}_i$, with $i \notin V^r_{it}$, can be tightened as $\mathrm{ES}'_i = \max\{\mathrm{ES}_i, \max_{\{h \in V^r_{it}:(h,i)\in A\}}(\mathrm{CF}_h - p^{\max}_h + \delta'_{hi})\}$ and $\mathrm{LS}'_i = \min\{\mathrm{LS}_i, \min_{\{h \in V^r_{it}:(i,h)\in A\}}(\mathrm{CF}_h - p^{\min}_h - \delta'_{ih})\}$, being $\mathrm{CF}_h$ the current finish time of the scheduled activity $h \in V^r_{it}$, where $\delta'_{hi}$ and $\delta'_{ih}$ are underestimations of $\delta_{hi}$ and $\delta_{ih}$, respectively. Consequently, $\mathrm{EF}_i$ and $\mathrm{LF}_i$ are updated as $\mathrm{EF}'_i = \mathrm{ES}'_i + p^{\min}_i$ and $\mathrm{LF}'_i = \mathrm{LS}'_i + p^{\max}_i$.

*Node selection* The algorithm selects the node with the largest lower bound. Ties are broken selecting the left-most node in the tree.

*Branching rule* Starting from the selected node $P_{it}$, the branching rule we adopt selects an unscheduled activity $i' \notin V^r_{it}$ so as to generate the largest increase of $u_{k\tau}$, with $k = 1, \ldots, K$, and $\tau = \mathrm{ES}'_{i'} + 1, \ldots, \mathrm{EF}'_{i'}$, when it is scheduled within $[\mathrm{ES}'_{i'} + 1, \mathrm{EF}'_{i'}]$ with the minimum duration. Then, children $P_{i't'}$ of $P_{it}$ are generated for each time period $t' \in \{\mathrm{EF}'_{i'}, \ldots, \mathrm{LF}'_{i'}\}$ by using $f_{i't'}$ as branching variable (see Fig. 1).

*Bounding phase at a generic node of the search tree* Let $\mathcal{UB}_{\text{best}}$ be the best objective function value found so far. Once an activity $i$ and a time period $t$ are selected by applying the branching rule, we have a new node $P_{it}$. Let $\mathcal{F}_{it}$ be the mathematical formulation of the subproblem related to node $P_{it}$: this formulation is obtained from $\mathcal{F}$ by fixing the values of the binary variables $f$ associated with the finish time periods in the partial schedule $F_{it}$. We apply the following bounding rules:

- we solve the linear relaxation of $\mathcal{F}_{it}$. Let $\mathcal{LRV}$ be the optimal solution value. If all the variables $s$ and $f$ are integer and $\mathcal{LRV} < \mathcal{UB}_{\text{best}}$ then update $\mathcal{UB}_{\text{best}} = \mathcal{LRV}$. Moreover, the subproblem can be closed. Otherwise,
- we solve the Lagrangian relaxation of $\mathcal{F}_{it}$ with multipliers $\tilde{\lambda}$. Let $\mathcal{LARV}$ be the optimal solution value. If (i) variables $s$ and $f$ are integer, (ii) constraints (19) are satisfied, (iii) $\tilde{\lambda}_{kt} = 0$ if the associated constraint (19) is not active, and (iv) $\mathcal{LARV} < \mathcal{UB}_{\text{best}}$ then update $\mathcal{UB}_{\text{best}} = \mathcal{LARV}$.

As for the root node, the solutions of both the two relaxed versions of $\mathcal{F}_{it}$ in the generic node $P_{it}$ are computed by means of a commercial solver.

*Fathoming rules at a generic node of the search tree* Consider the partial schedule $F_{it}$ at the current node $P_{it}$ of the search tree and let $\tau$ be the last time slot used in such a partial schedule.

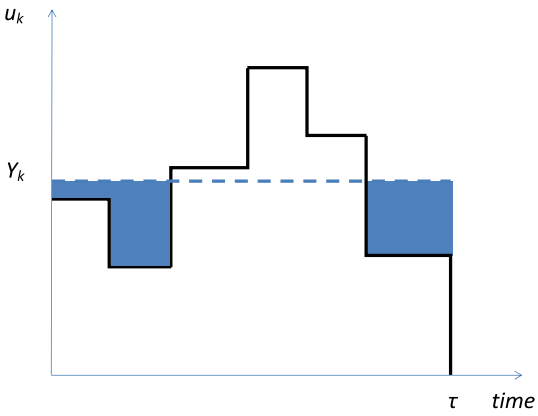


**Fig. 2** Example for $\mathcal{U}_{k\tau}$ and $\bar{\mathcal{U}}_{k\tau}$

1. Fathoming rule $FR_1$. The first rule is straightforward: if either $\mathcal{LRV} \geq \mathcal{UB}_{\text{best}}$ or $\mathcal{LARV} \geq \mathcal{UB}_{\text{best}}$, then we can prune the search tree.
2. Fathoming rule $FR_2$. Let

$$\mathcal{U}_{k\tau} = \sum_{t=1\ldots,\tau} \max(0, Y_k - u_{kt}),$$

that is, the unused area in the resource $k$ profile, measured from 1 to $\tau$ under the line $Y_k$, and let

$$\bar{\mathcal{U}}_{k\tau} = \sum_{t=1\ldots,\tau} \max(0, u_{kt} - Y_k),$$

that is, the used area above $Y_k$ from 1 to $\tau$ (see Fig. 2).
Moreover, let

$$\mathcal{N}_k = \sum_{i \notin V_{it}^r} \bar{r}_{ik},$$

that is, the overall resource type $k$ requirement for the activities not yet scheduled. Use the dark area in Fig. 2 to possibly satisfy such a resource requirement and let

$$\mathcal{R}_{k\tau} = \max\{0, \mathcal{N}_k - \mathcal{U}_{k\tau}\}$$

be the residual amount of $\mathcal{N}_k$.

If $\mathcal{R}_{k\tau} > 0$, fill the area with height equal to $Y_k$ and width equal to $(\bar{d} - \tau)$ with such a residual amount, as depicted in Fig. 3 (see the striped area).

Finally (see the dotted area depicted in Fig. 4), use the quantity

$$\bar{\mathcal{R}}_{k\tau} = \max\{0, \mathcal{R}_{k\tau} - Y_k \cdot (\bar{d} - \tau)\}$$

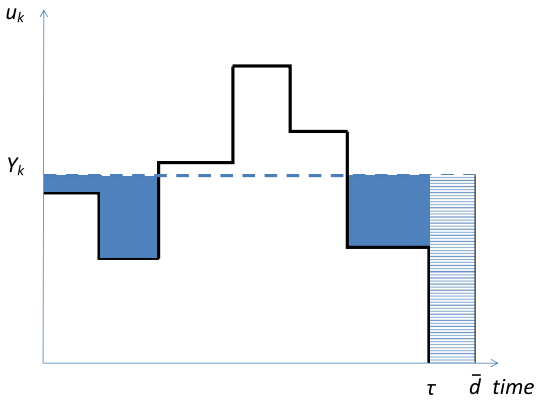


**Fig. 3** Example of filling area $Y_k \cdot (\bar{d} - \tau)$

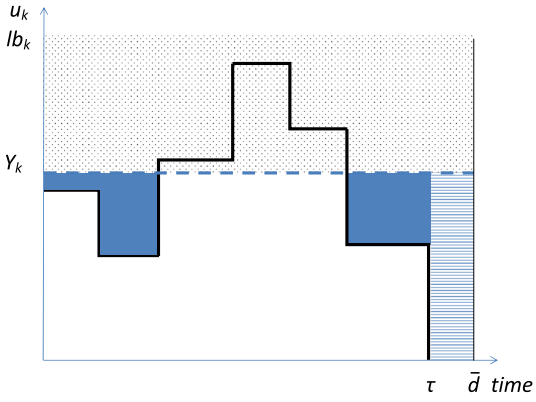


**Fig. 4** Example of lower bound $\text{lb}_k$

to calculate

$$\text{lb}_k = Y_k + \frac{\bar{\mathcal{R}}_{k\tau} + \bar{\mathcal{U}}_{k\tau}}{\bar{d}},$$

that is a lower bound on the minimum $u_{kt}$, with $t = 1, \ldots, \bar{d}$.
Now, we can state that

$$\mathcal{LB}(P_{it}) = \sum_{k=1}^{K} (\text{lb}_k - Y_k) \cdot c_k \cdot \bar{d}$$

is a lower bound on the objective function value of subproblem $P_{it}$, given the associated partial schedule $F_{it}$; therefore, if $\mathcal{LB}(P_{it})$ is greater than or equal to $\mathcal{UB}_{\text{best}}$ then the search tree can be pruned.

Once the algorithm has selected all the activities assigning a finishing time to each of them and an integer solution has not been found, it continues branching (i) using variable $s$ as branching variable, (ii) selecting activities in the same order as the one

used when branching variable $f$ was used, and (iii) assigning a start time to each activity.

## 5 Computational results

### 5.1 Presentation of the experiments and platform used

The algorithm has been implemented in both the C and AMPL languages. Software CPLEX 12.0 has been used to solve the relaxations of the mathematical formulations at the nodes of the branch and bound tree, i.e., those related to the calculations associated with the bounding phase at each node. The hardware used to carry out experiments is a Pentimum IV with 3 GHz clock and 2 GB RAM.

To test our algorithm we relied on two sets of known benchmarks. The first test set, denoted in the following as `TEST SET 1`, was introduced in Kolisch et al. (1999) and is available at.[1] Within this first set, we considered the collections denoted rlp_j10 and rlp_j20. The set rlp_j10 contains 270 instances each with ten activities. The number of renewable resources is 1, 3, and 5, but only one resource type is requested by each activity. The set rlp_j20 is formed of 270 instances with 20 activities and number of resources ranging from 1 to 5 as for the previous set. Out of these 270 instances we tested those ranging from 136 to 180 as done in Gather and Zimmermann (2009). All the data have been taken verbatim as reported in the library; the only exception is related to the duration. Indeed, since our problem is a generalization of the resource levelling problem to which the dataset used refers to, we had to make the following assumption: the duration reported in the instances has been assumed to be $p_i^{\max}$, while $p_i^{\min}$ has been chosen as $\lceil 0.75 \cdot p_i^{\max} \rceil$. Deadline $\bar{d}$ has been set equal to the length of the longest path between node 0 and node $n+1$ in the project network, assuming $p_i^{\max}$ as activity durations.

The second test set, denoted in the following as `TEST SET 2`, is available at.[2] These instances have been generated by the ProGen/max instance generator (Schwindt 1998) and are characterized by a restrictiveness of Thesen (measures the degree to which precedence constraints restrict the total number of feasible activity sequence) equal to 0.3 and 0.6. We considered instances with 10, 20, and 30 activities. Activities may require 1, 3, or 5 renewable resources. Differently from the previous dataset we notice that each activity may use simultaneously more than one resource type for its execution. Durations range from 1 to 10. As for `TEST SET 1`, the duration reported in the instances has been assumed to be $p_i^{\max}$, while $p_i^{\min} = \lceil 0.75 \cdot p_i^{\max} \rceil$. As for the previous test set, we pose $\bar{d}$ equal to the length of the longest path between node 0 and node $n+1$ in the project network, assuming $p_i^{\max}$ as activity durations.

[1] http://www.wiwi.tu-clausthal.de/en/chairs/produktion/research/research-areas/project-generator/rlpmax/.

[2] http://www.wiwi.tu-clausthal.de/abteilungen/unternehmensforschung/forschung/benchmark-instances/exact-results-for-single-mode-resource-levelling-problems/.

**Table 1** Computational results on TEST SET 1

| Test_Type | #_inst. | AOV | CPU_BB | #_opt_BB | #_opt_300 | CPU_CPLEX (s) | #_opt_CPLEX |
|---|---|---|---|---|---|---|---|
| rlp_j10 | 270 | 380 | 89 ms | 270 | 270 | 87 | 270 |
| rlp_j20 (inst. 136–180) | 45 | 805 | 2390 s | 42 | 18 | 5120 | 35 |

### 5.2 Results on TEST SET 1

In Table 1, we reported averages of the results obtained on TEST SET 1; we compared the results obtained by our algorithm with the results obtained by applying CPLEX on the mathematical formulation $\mathcal{F}$. We note that the machine used for the experiments has a single-core processor and that a multi-core environment may improve the performance of CPLEX. In the columns we listed:

- Test_Type the test type,
- #_inst. the number of instances of each test type,
- AOV the average objective value obtained by the proposed branch and bound,
- CPU_BB the average CPU time (in s) of our branch and bound to solve at the optimum the instances (when optimality is achieved),
- #_opt_BB the number of instances solved at the optimum within a time limit of 3 h by the proposed branch and bound,
- #_opt_300 the number of instances solved at the optimum within 300 s by our algorithm,
- #_opt_CPLEX the number of instances solved at the optimum within a time limit of 3 h by CPLEX,
- CPU_CPLEX the average CPU time (in s) of CPLEX to solve at the optimum the instances (when optimality is achieved).

Results reported in Table 1 show that our algorithm is able to solve instances with ten activities with a negligible computing time. As far as $n$ increases to 20 activities the CPU times increase significantly due to the theoretical complexity of the problem. In particular, we notice that 42 out of the 45 tested instances where solved at the optimum within 3 h of time limit with an average running time of 2390 s.

Looking at the results obtained by solving the mathematical formulation $\mathcal{F}$ on the same instances by means of the commercial solver CPLEX (see the last two columns of Table 1), we notice that, referring to instances with ten activities, even though all the 270 instances have been optimally solved as for our algorithm, computing times grow significantly from an average of less than 1 s to an average of 87 s. Moreover, when instances with 20 activities are concerned, not only computing times tend to increase (from an average of 2390 s to an average of 5120 s), but also the number of instances solved at the optimum reduces from 42 to 35. In the following, we analyze by means of charts the characteristics of the solutions obtained by our algorithm.

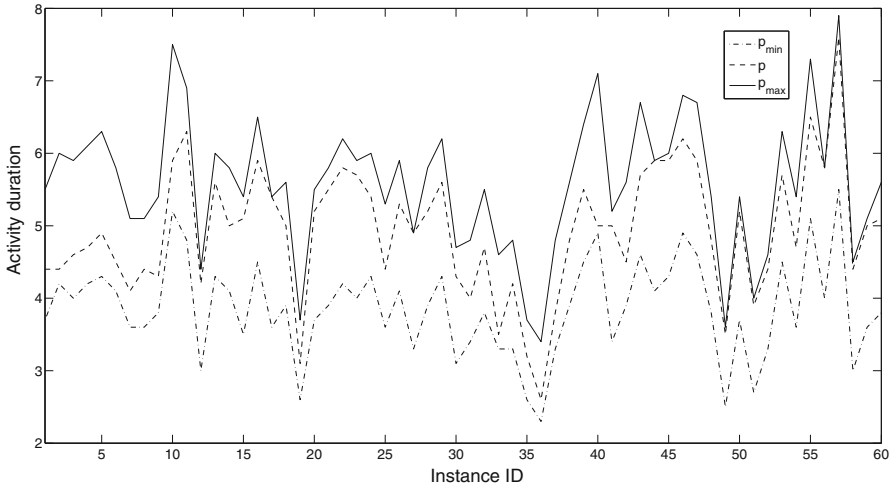


**Fig. 5** TEST SET 1: average $p_{\min}$, $p_{\max}$, and $p$ values for instances with ten activities (first 60 instances)

### *5.2.1 Average $p_{\min}$, $p_{\max}$, and $p$ values*

In Fig. 5, we plot the average activity duration $p$ obtained solving the instances with ten activities (for ease of presentation we report only the first 60 instances). We notice that these values are always strictly in between $p_{\min}$ and $p_{\max}$, being the latter the average $p_i^{\min}$ and $p_i^{\max}$ values over the number of activities. This behavior, i.e., $p$ never attains either $p_{\min}$ or $p_{\max}$, allows us to state that the model is in charge of using effectively the constraints on the minimum and maximum duration of each activity. An analogous behavior can be observed for the other 210 instances with ten activities, and also for the instances with 20 activities (see Fig. 6).

### *5.2.2 Average minimum and maximum variations of $x$ values*

In Fig. 7, we reported the minimum and maximum values, say $\Delta x_{\min}$ and $\Delta x_{\max}$, respectively, averaged over the number of activities, assumed by $\Delta x = x_{it} - x_{i,t-1}$ between the start and the end of activity $i$, for each instance ID. It is remarkable to note that $\Delta x_{\min}$ is never equal to $\Delta x_{\max}$, i.e, on average, activities are not scheduled uniformly over time. This means that the model is able to exploit the fractional $x$ variables to schedule activities choosing for each time period the best amount of resources to be allocated to each activity, and, in turn, the correct percentage of activity to be carried out. Figure 8 shows the same behavior on instances with 20 activities.

### *5.2.3 A further test with different values of $p_i^{\min}$*

We performed a further test on instances with 20 activities, reporting the objective function values obtained when $p_i^{\min}$ is taken equal to $p_i^{\max}$ (setting $p_i^{\min} = p_i^{\max}$, for all $i \in V$, leads to obtain the total overload cost problem). Figure 9 shows the

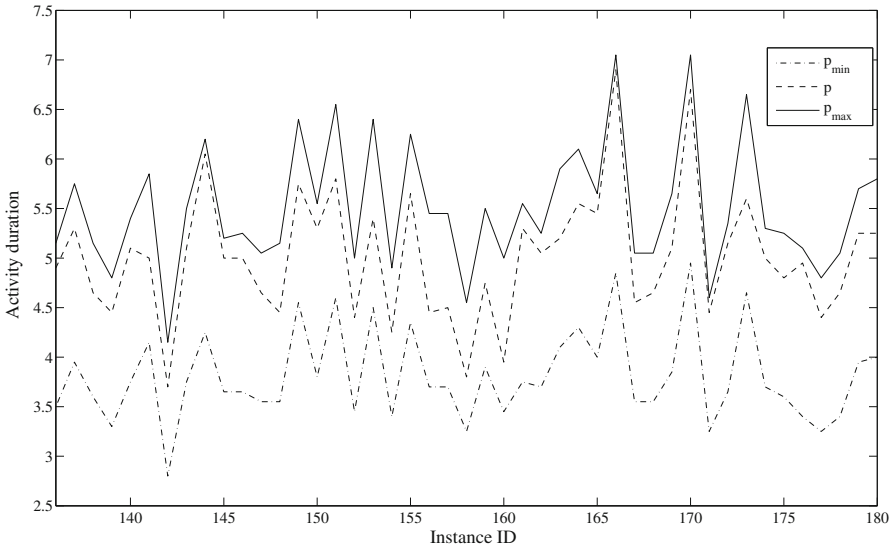


**Fig. 6** TEST SET 1: average $p_{\min}$, $p_{\max}$, and $p$ values for instances with 20 activities

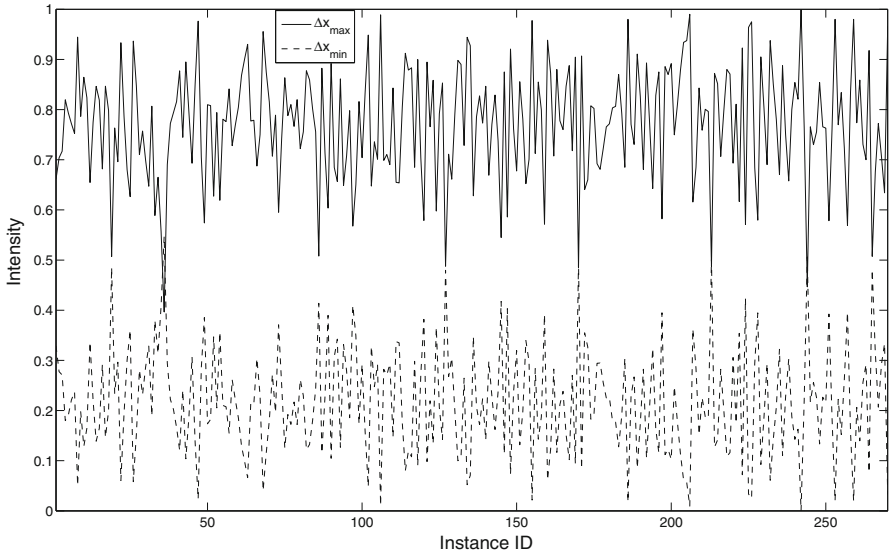


**Fig. 7** TEST SET 1: average minimum and maximum $\Delta x$ values for instances with ten activities

comparison of the objective values of the latter scenario with respect to the scenario with $p_i^{\min} = \lceil 0.75 \cdot p_i^{\max} \rceil$. We note that there exist instances for which the objective values of the scenario with $p_i^{\min} = \lceil 0.75 \cdot p_i^{\max} \rceil$ are better than the objective values of the scenario with $p_i^{\min} = p_i^{\max}$, which means that the model is able to exploit the variability of the durations, leveling the resource profiles over time more effectively.

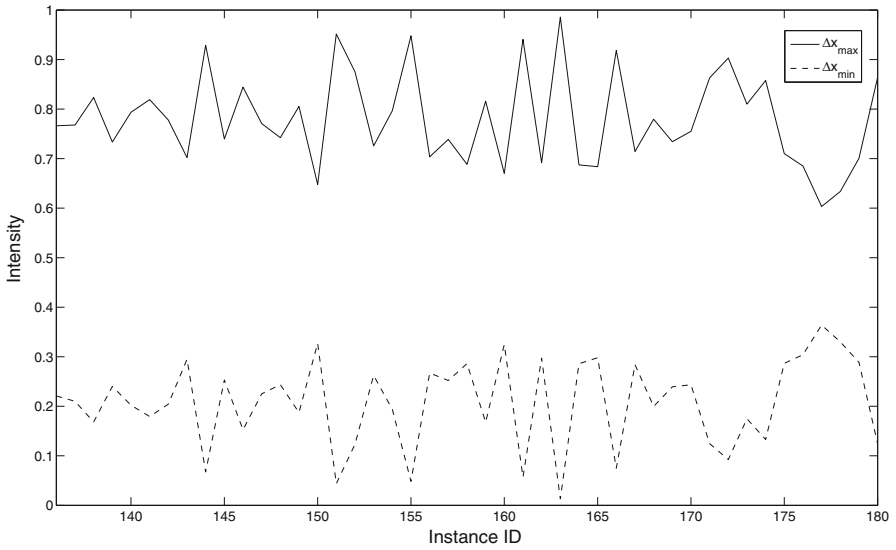


**Fig. 8** TEST SET 1: average minimum and maximum $\Delta x$ values for instances with 20 activities

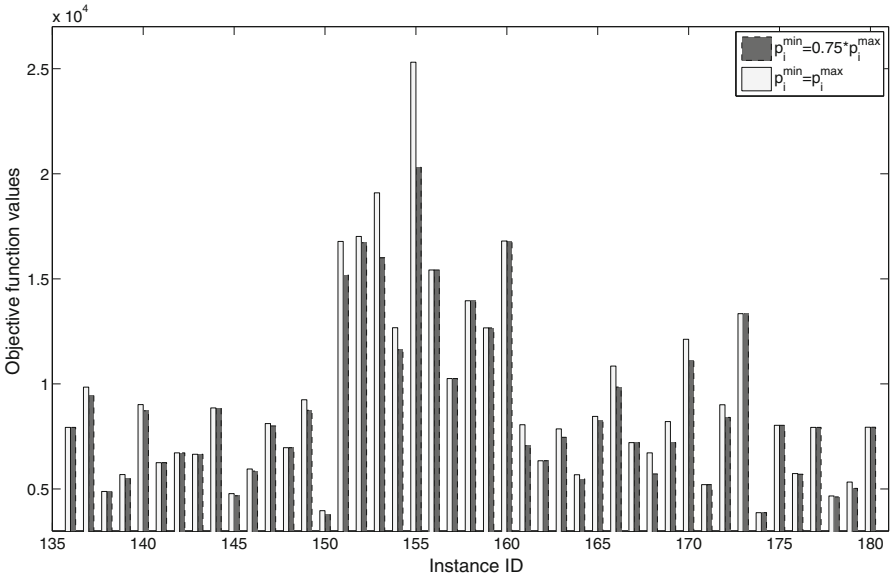


**Fig. 9** TEST SET 1: comparison of objective function values obtained with two different values of $p_i^{\min}$ on instances with 20 activities

### 5.3 Results on TEST SET 2

In Tables 2 and 3, we report average results obtained on TEST SET 2; as for TEST SET 1, we compared the results obtained by our algorithm with the results obtained by applying CPLEX on the mathematical formulation $\mathcal{F}$. In the columns we listed:

- Test_Type the test type, denoted with $n\_K$ where $n$ is the number of real activities and $K$ is the number of renewable resources,
- #_inst. the number of instances of each test type,
- AOV the average objective value obtained by the proposed branch and bound,
- CPU_BB the average CPU time (in s) of our branch and bound algorithm to solve at the optimum the instances (when optimality is achieved),
- #_opt_BB the number of instances solved at the optimum within 3 h by our algorithm,
- #_opt_300 the number of instances solved at the optimum within 300 s by our algorithm,
- #_opt_CPLEX the number of instances solved at the optimum within a time limit of 3 h by CPLEX,
- CPU_CPLEX the average CPU time (in s) of CPLEX to solve at the optimum the instances (when optimality is achieved),
- $\Delta x_{\min}$ the average minimum activity intensity in the optimal solutions,
- $\Delta x_{\max}$ the average maximum activity intensity in the optimal solutions,
- $p_{\min}$ the average minimum duration allowed,
- $p$ the average duration used in the optimal solutions,
- $p_{\max}$ the average maximum duration allowed.

By the results reported in Table 2, we notice that our algorithm is able to solve very efficiently instances with ten activities for different values of $K$ as it happens for instances comprised in TEST SET 1. The running times, as expected, tend to increase rapidly as $n$ grows: experimental records say that 90 % of the 120 instances with 20 activities have been solved optimally with an average CPU time of 2230 s; while 45 % of the 120 instances with 30 activities were optimally solved within an average time of 6138 s. These numbers appear to be comparable with the tree-based approach proposed in Gather et al. (2011) for the total overload cost problem without variable intensities and flexible durations, where all the instances with ten activities have been solved at the optimum within negligible computing times, and 87 % of the instances with 20 activities have been solved at the optimum with an average CPU time of 2184 s on an Intel Quad Core PC. Tests on 30 activity instances have not been reported.

Looking at the results obtained by solving the mathematical formulation $\mathcal{F}$ on the same instances by means of the commercial solver CPLEX (see the last two columns of Table 2), all the 120 instances with ten activities have been optimally solved, as for our algorithm, with computing times significantly larger since they raise from an average of less than 1 s used by our approach to an average of 91 s. Moreover, when instances with 20 activities are considered, computing times grow from an average of 2230 s to an average of 5411 s, and the number of optimally solved instances reduces from 108 (90 %) to 87 (72.5 %). Finally, tests on instances with 30 activities show that CPLEX is able to optimally solve 37 instances, as opposed to the 54 solved by our algorithm, with an average computing time of 7916 s compared to the 6138 s used by our algorithm.

Table 3 reports values on average minimum and maximum $\Delta x$ values and durations. The results, as those discussed by means of charts for TEST SET 1, suggest that the

**Table 2** Computational results on TEST SET 2

| Test_Type | #_inst. | AOV | CPU_BB | #_opt_BB | #_opt_300 | CPU_CPLEX (s) | #_opt_CPLEX |
|---|---|---|---|---|---|---|---|
| 10_1 | 40 | 69.25 | 50 ms | 40 | 40 | 82 | 40 |
| 10_3 | 40 | 210.15 | 79 ms | 40 | 40 | 90 | 40 |
| 10_5 | 40 | 382.45 | 98 ms | 40 | 40 | 101 | 40 |
| 20_1 | 40 | 132.85 | 1429 s | 38 | 0 | 3442 | 31 |
| 20_3 | 40 | 425.90 | 2194 s | 38 | 0 | 5112 | 30 |
| 20_5 | 40 | 689.25 | 3223 s | 32 | 0 | 8102 | 26 |
| 30_1 | 40 | 198.35 | 3842 s | 21 | 0 | 6018 | 15 |
| 30_3 | 40 | 684.50 | 5225 s | 18 | 0 | 8144 | 12 |
| 30_5 | 40 | 1025.15 | 10,448 s | 15 | 0 | 10,489 | 10 |

**Table 3** TEST SET 2: average $\Delta x_{\min}$, $\Delta x_{\max}$, $p_{\min}$, $p$, and $p_{\max}$ values

| Test_Type | $\Delta x_{\min}$ | $\Delta x_{\max}$ | $p_{\min}$ | $p$ | $p_{\max}$ |
|---|---|---|---|---|---|
| 10_1 | 0.23 | 0.59 | 3.51 | 4.24 | 5.34 |
| 10_3 | 0.25 | 0.60 | 3.62 | 4.25 | 5.35 |
| 10_5 | 0.27 | 0.62 | 3.72 | 4.28 | 5.42 |
| 20_1 | 0.29 | 0.62 | 3.59 | 4.26 | 5.32 |
| 20_3 | 0.30 | 0.64 | 3.60 | 4.29 | 5.38 |
| 20_5 | 0.31 | 0.66 | 3.73 | 4.30 | 5.44 |
| 30_1 | 0.33 | 0.67 | 3.66 | 4.23 | 5.34 |
| 30_3 | 0.30 | 0.69 | 3.75 | 4.28 | 5.48 |
| 30_5 | 0.28 | 0.70 | 3.81 | 4.32 | 5.50 |

**Table 4** TEST SET 2: average AOV for instances with $p_i^{\min} = \lceil 0.75 \cdot p_i^{\max} \rceil$ ($\text{AOV}_{\text{diff}}$) and $p_i^{\min} = p_i^{\max}$ ($\text{AOV}_{\text{eq}}$)

| Test_Type | $\text{AOV}_{\text{diff}}$ | $\text{AOV}_{\text{eq}}$ |
|---|---|---|
| 10_1 | 71.28 | 78.48 |
| 10_3 | 215.85 | 240.88 |
| 10_5 | 375.50 | 423.85 |
| 20_1 | 125.45 | 157.05 |
| 20_3 | 402.8 | 457.90 |
| 20_5 | 636.5 | 768.00 |
| 30_1 | 180.25 | 220.00 |
| 30_3 | 656.85 | 769.90 |
| 30_5 | 782.25 | 1190.48 |

proposed model is capable of exploiting its features. Indeed, to have a better idea of the gain produced by using the proposed model we made a further test, similarly to what done for TEST SET 1, with $p_i^{\min} = p_i^{\max}$ and fixed execution intensities equal to $\frac{1}{p_i^{\min}} = \frac{1}{p_i^{\max}}$. The results, reported in Table 4, show that for instances with ten activities the average objective values are, on average, 12 % larger, for instances with 20 activities this gap increases to 18 %, and, for instances with 30 activities it reaches 37 %.

## 6 Conclusions

In this paper, we studied the problem of levelling resources in a project (focusing on the total overload cost version) with generalized precedence relationships, given a deadline for the completion of all the activities and variable execution intensity and duration of the activities. To the best of our knowledge, generalized precedence relationships and variable intensity execution and duration have not been dealt with together. For this novel problem we proposed a mixed-integer linear programming formulation, a lower bound based on Lagrangian relaxation, and a branch and bound algorithm.

Computational results on known benchmarks were provided and highlighted how the proposed model is able to manage all the considered ingredients in such a way that produces promising solutions.

## References

Ahuja HN (1976) Construction performance control by networks. Wiley, New York
Bartusch M, Möhring RH, Radermacher FJ (1988) Scheduling project networks with resource constraints and time windows. Ann Oper Res 16(1–4):201–240
Bianco L, Caramia M (2011) Minimizing the completion time of a project under resource constraints and feeding precedence relations: a Lagrangian relaxation based lower bound. 4OR 9:371–389
Bianco L, Caramia M (2012) An exact algorithm to minimize the makespan in project scheduling with scarce resources and generalized precedence relations. Eur J Oper Res 219(1):73–85
Burgess AR, Killebrew JB (1962) Variation in activity level on a cyclical arrow diagram. Int J Ind Eng 2:76–83
Easa SM (1989) Resource levelling in construction by optimization. J Constr Eng Manag 115:302–316
Engelhardt H, Zimmermann J (1998) Lower Bounds and exact methods for resource levelling problems. Technical Report WIOR (Universität Karlsruhe), vol 517
Fündeling C, Trautmann N (2010) A priority-rule method for project scheduling with work-content constraints. Eur J Oper Res 203:568–574
Gabow HN, Myers EW (1978) Finding all spanning trees of directed and undirected graphs. SIAM J Comput 7:280–287
Gather T, Zimmermann J (2009) Exact methods for the resource levelling problem. Proc MISTA Conf 2009:811–820
Gather T, Zimmermann J, Bartels J-H (2011) Exact methods for the resource levelling problem. J Sched 14(6):557–569
Kis T (2005) A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. Math Program 103(3):515–539
Kolisch R, Schwindt C, Sprecher A (1999) Benchmark instances for project scheduling problems. In: Weglarz J (ed) Project scheduling: recent models, algorithms, and applications, vol 9. Kluwer, Boston, pp 197–212
Kreter S, Rieck J, Zimmermann J (2014) The total adjustment cost problem: applications, models, and solution algorithms. J Sched 17:145–160
Neumann K, Schwindt C, Zimmerman J (2003) Project scheduling with time windows and scarce resources. In: Lecture notes in economics and mathematical systems, vol 508, 2nd edn. Springer, Berlin
Nübel H (2001) The resource renting problem subject to temporal constraints. OR Spektrum 23:359–381
Petrovic R (1969) On optimization of resource leveling in project plans. In: Lombaers HJ (ed) Project planning by network analysis. North-Holland, Amsterdam, pp 268–273
Rieck J, Zimmerman J (2015) Exact methods for resource leveling problems. In: Schwindt C, Zimmerman J (eds) Handbook on project management and scheduling, vol 1. Springer, Berlin, pp 361–387
Rieck J, Zimmerman J, Gather T (2012) Mixed-integer linear programming for resource levelling problems. Eur J Oper Res 221:27–37
Schwindt C (1998) Verfahren zur Lösung des ressourcenbeschränkten Projektdauerminimierungsproblems mit planungsabhängigen Zeitfenstern. Shaker, Aachen
Schwindt C, Zimmerman J (2015) Handbook on project management and scheduling, vol 1. Springer, Berlin