OR Spectrum (2020) 42:901–933
https://doi.org/10.1007/s00291-020-00595-9

REGULAR ARTICLE

The multi-mode resource investment problem: a benchmark library and a computational study of lower and upper bounds

Patrick Gerhards[1]

Received: 10 April 2019 / Accepted: 25 June 2020 / Published online: 21 July 2020

Abstract

The multi-mode resource investment problem (MRIP) is the multi-mode extension of the resource investment problem, which is also known under the name resource availability cost problem. It is a project scheduling problem with a given due date as well as precedence and resource constraints. The goal is to find a precedence feasible schedule that minimises the resource costs of the allocated resources. To compute these costs, the maximum resource peak is considered regarding renewable resource types, whereas the sum of allocated nonrenewable resource units is used in the case of nonrenewable resources. Many practical and complex project scheduling settings can be modelled with this type of problem. Especially with the usage of different processing modes, time and cost compromises can be utilised by the project manager. In the literature, some procedures for the MRIP have been investigated; however, the computational experiments in these studies have not been carried out on common benchmark instances. This makes a fair comparison of methods difficult. Therefore, we generated novel instances specifically designed for this problem and published them on the website https://riplib.hsu-hh.de. On this website, the instances as well as best-known solution values are available and researchers can also contribute their findings. We investigate these novel instances by proposing and evaluating lower bounds for the MRIP. Additionally, we analyse the proposed instances by applying heuristic as well as exact methods. These experiments suggest that most of the instances are challenging and further research is needed. Finally, we show some computational complexity properties of the NP-hard MRIP.

Keywords Project scheduling · Resource investment problem · Multi-mode · Benchmark instances · Mixed-integer programming

✉ Patrick Gerhards
patrick.gerhards@hsu-hh.de

[1] Helmut Schmidt University, Hamburg, Germany

1 Introduction

Project scheduling is an important part of project management. Often, projects have a deadline or a due date and the tasks of the project compete against each other in the use of scarce resources. In the resource investment problem (RIP), the decision maker can determine how many units of each resource are allocated to the project in order to find a schedule that meets the due date. This schedule needs to respect the allocated resources as well as the precedence relations among the activities of the project. The goal is to minimise the project costs that are defined by the allocated units of resources. In this work, we consider two types of resources: renewable and nonrenewable resources. The amount of available renewable resource units is replenished after each time period. So, for the cost calculation of the renewable resource costs, the maximum resource usage peak is multiplied with a cost factor. This cost factor defines the costs of adding an extra resource unit to the project that is available in each period of the project horizon. Nonrenewable resource units are consumed for the entire project horizon and their resource costs are calculated by considering the total sum of consumed resource units.

The resource investment problem was firstly introduced by Möhring (1984) and is also known as the resource availability cost problem (RACP) in the literature. Möhring (1984) describes it as a "problem of scarce time" in contrast to the well-known and related resource-constrained project scheduling problem (RCPSP) which is a "problem of scarce resources". With the RIP, it is possible to model a wide range of applications such as the construction or dismantling of buildings or software development projects, just to name a few (e.g. Bartels 2009). Several extensions of the problem were introduced in the literature. In this work, we consider the multi-mode resource investment problem (MRIP), which was firstly introduced by Hsu and Kim (2005). Here, each activity can be processed in multiple modes which vary in the resource consumption and the processing duration. For the RCPSP and its multi-mode extension, several benchmark instance libraries, such as the Boctor datasets (Boctor 1993), the PSPLIB (Kolisch and Sprecher 1997), or the MMLIB (Van Peteghem and Vanhoucke 2014), exist. Yet, for the MRIP, no benchmark instances are available to the public which makes the comparison of solution procedures hard. For this reason, we designed a set of benchmark instances and made them available to the public on the website https://riplib.hsu-hh.de.

The contributions of this work are the following:

- We show that it is sufficient to consider only one nonrenewable resource in the MRIP and provide a transformation for instances with multiple nonrenewable resources.
- We describe and compare different approaches to compute lower bounds for the MRIP.
- We have set-up and maintain a website https://riplib.hsu-hh.de. Here, researchers can access a new set of benchmark instances for the MRIP and share their results with others.

- We use different exact and heuristic procedures to solve the novel benchmark instances. This enables us to gain insights into which characteristics lead to "easy" and "hard" instances.

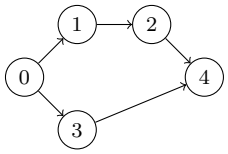
This paper is organised as follows: In Sect. 2, we give a formal definition of the problem and show that we can aggregate multiple nonrenewable resources into a single one. Section 3 provides an overview of the existing literature concerning the RIP or extensions of it. Next, we present several lower bound procedures as well as different optimisation techniques to generate upper bounds for the MRIP in Sect. 4. We applied mixed-integer programming (MIP), constraint programming (CP), simulated annealing (SA) and a multi-start local search (MLS). In Sect. 5, we explain which parameters we used for generating benchmark instances and what difficulties we encountered in the process of doing so. Sect. 6 contains a computational study with our findings considering the performance of the procedures presented in Sect. 4. Finally, in Sect. 7, we finish this work with a critical appraisal and an outlook for further research.

2 Problem statement

An instance of the MRIP is defined by the following properties: a set of activities $A = \{0, \dots, n+1\}$, a set of precedence relations $E \subseteq A \times A$, a set of renewable resources $\mathcal{R}$ and a set of nonrenewable resources $\mathcal{R}^n$. For each activity, $i \in A$, a set of modes M_i exists and for each mode $m \in M_i$, the duration $d_{im} \in \mathbb{Z}_0^+$ is given. Activity 0 and $n+1$ are dummy activities that mark the beginning and end of the project and their duration and resource requirements are equal to 0. In this work, we consider only finish-to-start precedence constraints among activities which are represented by the set E. So, if $(i, j) \in E$ for activities $i, j \in A$ this means that there is a minimum time lag of 0 between the finish of activity i and the start of activity j, i.e. j can only start after i is finished.

The due date $D \in \mathbb{Z}^+$ is fixed and defines the maximum project duration. The resource requirement $r_{imk} \in \mathbb{Z}_0^+$ ($r^n_{imk} \in \mathbb{Z}_0^+$) of each non-dummy activity i depends on the mode m and the renewable resource $k \in \mathcal{R}$ (nonrenewable resource $k \in \mathcal{R}^n$). For each renewable resource $k \in \mathcal{R}$ (nonrenewable resource $k \in \mathcal{R}^n$) a resource cost factor $c_k \in \mathbb{Z}_0^+$ ($c^n_k \in \mathbb{Z}_0^+$) defines the price of allocating one unit of the resource to the project. For renewable resources, the allocated amount of the resource is replenished after each time period. A schedule is resource feasible with respect to the renewable resources if the amount of allocated resource units is larger than or equal to the maximum resource peak usage. That means that for each period of the project horizon, the sum of resource requirements of activities that are in process in this period has to be smaller than or equal to the allocated amount. To calculate the renewable resource costs, we multiply the allocated amount that corresponds with the maximum resource peak with the resource cost factor. The renewable resource type is useful to model for example workers or machines.

The amount of allocated nonrenewable resources, however, is consumed by the activities over the whole project and they are not replenished. To calculate the

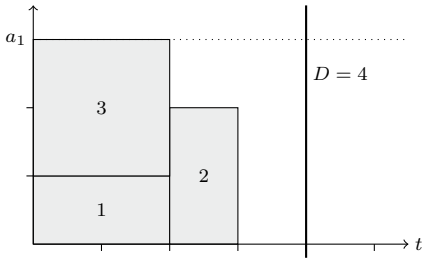


(a) Example activity-on-the-node network

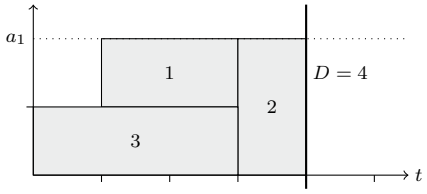
i	m	d_{im}	$r_{i,m,1}$	$r^n_{i,m,1}$
0	1	0	0	0
1	1	2	1	0
2	1	1	2	0
3	1	2	2	0
	2	3	1	1
4	1	0	0	0

(b) Duration and resource consumption

Fig. 1 Illustrative example data



(a) Example schedule for $D = 4$



(b) Example schedule with different mode for activity 3

Fig. 2 Example schedule

nonrenewable resource costs, we sum up the nonrenewable resource requirements of all activities and multiply them with the respective cost factor (i.e. no peak usage is considered as the resource units do not replenish here). Resources of this type can represent budgets or rare materials. The nonrenewable resources are also useful to model outsourcing certain activities to external contractors.

A small example of an MRIP instance is depicted in Fig. 1. Here, we have five activities with activities 0 and 4 being the dummy start and dummy end activity of the project, respectively. The arcs in the network represent the precedence relations among activities. So, e.g. activity 2 can only start after activity 1 is finished. For each activity, there is only one mode available except for activity 3. Fig. 1b depicts the duration of each mode as well as the renewable resource consumption $r_{i,m,1}$ and the nonrenewable resource consumption $r^n_{i,m,1}$. Hence, we consider only one renewable and one nonrenewable resource in this example. The unit cost factors are $c_1 = 2$ and $c^n_1 = 1$ and there is no upper bound on the capacity of each resource. The due date of the project is $D = 4$. In Fig. 2a, we show a Gantt chart, where each activity is processed in mode 1. Since there is a renewable resource usage of 3 units in period 1 and 2, we need to allocate 3 units of the renewable resource to the project. Because of the mode choice, 0 units of the nonrenewable resource are utilised, and thus, the schedule has a cost value of 6. The precedence relations as well as the due date are respected. By switching the mode of activity 3 (mode 2 instead of mode 1), we obtain a better solution (Fig. 2b). Here, only 2 units of the renewable resource are allocated to the project since the peak resource

usage is now 2 instead of 3, but also 1 nonrenewable resource unit is needed. Therefore, the cost of the schedule depicted in Fig. 2b is 5.

In equations (1)–(7), we display a mathematical model formulation for the MRIP, which is also used in Sect. 4 in a MIP procedure. It is an adaptation of the model of Talbot (1982) for the multi-mode extension of the RCPSP. We use two types of decision variables in the mathematical model: real-valued variables a_k and a_k^n for the resource allocation of resource $k \in \mathcal{R} \cup \mathcal{R}^n$ and binary variables x_{imt} for the mode and scheduling choice. The variable x_{imt} takes a value of 1 if and only if activity i is processed in mode $m \in M_i$ and starts at period t. For the start period, we compute lower (and upper) bounds called ES_i (LS_i) using forward and backward calculation (FBC, similar to Kelley 1963). Here, we use the minimum duration of each activity and the due date D acts as an upper bound of the latest start time of the project dummy end activity $n+1$ (i.e. $LS_{n+1} = D$). We also compute an upper bound LF_i on the latest finish period with respect to the due date D using backward calculation, and hence, $LF_i - d_{im}$ is the latest possible start of activity i if it is processed in mode m.

In the objective function (1), the resource costs are minimised. Constraints (2) enforce that for each activity i exactly one mode and one start period is chosen. The inequalities (3) represent the precedence relations: if $(i,j) \in E$, then the finish period of activity i (left side of the inequality) has to be lower than or equal to the start period of activity j (right side). Inequalities (4) and (5) make sure that the amount of allocated resources a_k^n and a_k is as least as high as the consumption of the nonrenewable and renewable resources, respectively. Finally, in terms (6) and (7), the two different types of decision variables are depicted. According to Artigues (2017), the binary variables are so-called *pulse* variables over discrete time periods and inequalities (3) are the so-called *aggregated* precedence constraints. Hence, we call the formulation displayed in (1)–(7) the "aggregated discrete time formulation based on pulse variables" (PDT). In Sect. 4, we will use this formulation and others (using the disaggregated version of the precedence constraints and/or other decision variable types) in a MIP.

$$\min \sum_{k \in \mathcal{R}} c_k \cdot a_k + \sum_{k \in \mathcal{R}^n} c_k^n \cdot a_k^n \tag{1}$$

$$s.t. \sum_{m \in M_i} \sum_{t=ES_i}^{LF_i - d_{im}} x_{imt} = 1 \quad \forall i \in A \tag{2}$$

$$\sum_{m \in M_i} \sum_{t=ES_i}^{LF_i - d_{im}} x_{imt}(t + d_{im}) \leq \sum_{m \in M_j} \sum_{t=ES_j}^{LF_j - d_{jm}} x_{jmt} \cdot t \qquad \forall (i,j) \in E \tag{3}$$

$$\sum_{i \in A} \sum_{m \in M_i} \sum_{t=ES_i}^{LF_i - d_{im}} x_{imt} \cdot r_{imk}^n \leq a_k^n \quad \forall k \in \mathcal{R}^n \tag{4}$$

$$\sum_{i \in A} \sum_{m \in M_i} \sum_{q=\max(ES_i, t-d_{im}+1)}^{\min(t, LF_i - d_{im})} x_{imq} \cdot r_{imk} \leq a_k \quad \forall k \in \mathcal{R}, t = 0, \ldots, D \tag{5}$$

$$a_k \geq 0, \quad a_k^n \geq 0 \quad \forall k \in \mathcal{R}^n \tag{6}$$

$$x_{imt} \in \{0, 1\} \quad \forall i \in A, \forall m \in M_i, t = ES_i, \ldots, LS_i. \tag{7}$$

Since the resource investment problem with a single mode per activity is NP-hard (cf. Möhring 1984), the multi-mode extension is also NP-hard. This is still true if we extend the problem setting with nonrenewable resources. However, it is sufficient to consider only a single nonrenewable resource since we can aggregate multiple nonrenewable resources into a single one. We compute the novel resource requirement of an activity i and a mode m to be the sum of the old resource requirements times the respective resource cost factor:

$$\overline{r}_{i,m,1}^n = \sum_{k \in \mathcal{R}^n} c_k^n \cdot r_{imk}^n. \tag{8}$$

The cost factor of this single nonrenewable resource is equal to 1. Note that this is possible since there is no upper bound on the resource use and the resource consumption is not time dependent for nonrenewable resources. For this aggregated nonrenewable resource, we can set the allocation to be $\overline{a}_1^n = \sum_{k \in \mathcal{R}^n} c_k^n \cdot a_k^n$ based on allocations a_k^n of the former resources. Obviously, the objective value does not change by this aggregation and also all constraints such as precedence relations and renewable resource constraints are not changed. This transformation is the reason why we consider only instances with a single nonrenewable resource in this study and it can be used to convert instances with multiple nonrenewable resources into the single resource case.

3 Literature review

The RIP is closely related to other project scheduling problems such as the RCPSP or the resource levelling problem (RLP). However, the goal of the RCPSP and its multi-mode extension (MRCPSP) is the minimisation of the makespan with fixed resource availabilities. Several heuristic and exact procedures have been proposed for the MRCPSP (e.g. Geiger 2017 and Schnell and Hartl 2016, respectively). In the RLP, also a due date for the latest project completion is given, yet the objective function often differs. Several resource levelling objective functions are known in the literature such as the total squared utilisation cost or the total overload cost (Rieck and Zimmermann 2015). Bianco et al. (2016) extended the RLP setting with generalised precedence relations and variable intensities in the execution of the activities. Another closely related problem to the RIP is the time-constrained project scheduling problem (TCPSP). It can be seen as a combination of the RIP and the RCPSP since there is a given due date as well as resource capacities. However, additional

capacities can be temporarily allocated to the project at a certain cost. So, it has to be decided in which periods extra resource units are added. The goal is to minimise the cost for additional resources under the given due date. This problem was firstly proposed by Deckro and Hebert (1989), and lately, Verbeeck et al. (2017) proposed an artificial immune system (AIS) implementation for the TCPSP.

Next, we give a brief literature overview on the existing work for the RIP and some of its extensions. Several exact solution methods for the RIP exist. Möhring (1984) was the first to use the RIP to model a bridge construction project. He proposed an exact method using graph theoretical algorithms to solve the problem. Another exact algorithm, called minimum bounding algorithm (MBA), was introduced by Demeulemeester (1995), where a branch-and-bound procedure for the RCPSP is applied iteratively. Improving this procedure, Rodrigues and Yamashita (2010, 2015) proposed the modified minimum bounding algorithm by utilising a feasible initial solution that is found heuristically. Lately, Kreter et al. (2018) studied the RIP as well as an extension with general temporal constraints and one with calendar constraints. They provided mixed-integer linear programming formulations as well as constraint programming (CP) implementations for the problems. With their CP procedure, they were able to close all available instances from the literature for these single-mode problems. The authors also give an overview on previous work on the RIP with generalised precedence constraints (RIP/max). On their website, the authors provide instances for the single-mode RIP as well as the RIP/max with up to 500 activities per project.

Drexl and Kimms (2001) studied the computation of lower bounds (LB) for the RIP. They proposed two procedures: one using Lagrangian relaxation and the other based on column generation techniques. Both procedures also yield feasible solutions for the problem as a by-product. The authors conducted computational experiments on project instances with up to 30 activities and up to 8 resources.

The initial application of a metaheuristic for the RIP was proposed by Yamashita et al. (2006). They implemented a scatter search (SS) which is a population-based metaheuristic and it outperformed two simple multi-start heuristics as well as the upper bounds obtained by Drexl and Kimms (2001) on instances with up to 120 activities. Ranjbar et al. (2008) also implemented population-based metaheuristics: path relinking (PR) and genetic algorithm (GA). Here, the PR achieved slightly better results than the GA on the instances generated in Yamashita et al. (2006), but due to different hardware they did not compare their results directly to the SS mentioned above. An implementation of the AIS metaheuristic as well as a benchmark set with 30 activities and 4 resources (called RACP30) is presented by Van Peteghem and Vanhoucke (2013). They show that their approach outperforms the GA for the tardiness permitted extension presented by Shadrokh and Kianfar (2007) and the proposed instances are also used in Van Peteghem and Vanhoucke (2015). Meng et al. (2016) proposed a hybrid metaheuristic by combining the tabu search (TS) metaheuristic with the SS metaheuristic. They tested their procedure, called tabued scatter search (TSS), on 48 adapted RCPSP instances from the PSPLIB. Experiments showed that on average TSS is able to obtain better solutions than the SS of Yamashita et al. (2006), but by spending a higher computational time. A novel heuristic approach called multi-start iterative search heuristic (MSIS) is proposed

by Zhu et al. (2017). Here, the authors combine local search techniques with path relinking and show that their method outperforms the PR, the GA and the SS procedures presented earlier. The computational experiments were performed on adapted PSPLIB instances and newly generated ones.

Shadrokh and Kianfar (2007) began studying the extension of the RIP, where exceeding the due date (tardiness) is permitted but penalised in the objective function. The penalty costs rise for each time period that the project finishes later than its specified due date by a fixed tardiness cost factor. It is called resource investment problem with tardiness (RIPT) and the authors applied a GA to tackle this novel problem. Van Peteghem and Vanhoucke (2015) apply a metaheuristic procedure called invasive weed optimisation algorithm (IWO) to the RIP as well as the RIPT. Computational experiments performed on the RACP30 instance set as well as on 30 activity projects adapted from the PSPLIB indicate that the IWO outperforms both the AIS from Van Peteghem and Vanhoucke (2013) and the GA from Shadrokh and Kianfar (2007). Recently, Yuan et al. (2017) use the so-called moving block sequence (MBS) representation in an evolutionary algorithm (EA) for the RIPT. The authors conducted experiments on instances with up to 20 activities and results indicated that their procedure outperforms the GA of Shadrokh and Kianfar (2007).

The multi-mode extension of the RIP was firstly studied Hsu and Kim (2005). They proposed a heuristic procedure that combines two priority rules: one regarding the increase in costs when adding an activity to a partial schedule and the other considering how the finish time of the current activity affects its successors' remaining start times. They tested different weight combinations of this combined priority rule heuristic against heuristics that applied different priority rules sequentially (one to select the next activity and another one to select the mode and start time). For the experiments, they used MRCPSP instances from the PSPLIB with 12 to 30 activities, but treated the nonrenewable resources as renewable resources. Another heuristic procedure for the MRIP was presented by Qi et al. (2015). The authors proposed a novel schedule generation scheme as well as the modified particle swarm optimisation (MPSO) metaheuristic, which is a combination of particle swarm optimisation (PSO) and SS. They also adapted PSPLIB instances for the MRCPSP with 12–30 activities to test their MPSO heuristic against a PSO implementation and an adaptation of the GA of Ranjbar et al. (2008). Colak and Azizoglu (2014) proposed a heuristic procedure for the special case of the MRIP with a single renewable resource. Their approach uses different construction procedures and tries to improve a solution using several neighbourhood search strategies. The authors performed experiments on instances with up to 100 activities and up to 10 modes per activity.

For the MRIP, two exact procedures are known. Yamashita and Morabito (2009) combined the MBA of Demeulemeester (1995) with an exact branch-and-bound algorithm for the MRCPSP of Sprecher and Drexl (1998). In order to compute time/cost trade-off curves, they investigated several different due dates per instance. Since the procedure relies on solving multiple MRCPSP exactly, they solved only small instances with 15 activities per project. The other exact procedure was presented by Coughlan et al. (2015). The authors also added calendar constraints to the MRIP setting, making some resources not available at certain times. They proposed a Dantzig–Wolfe reformulation in combination with a column generation

and a branch-and-price algorithm, where they heavily utilise the calendar structure of the resource availability. In computational experiments, Coughlan et al. (2015) compared their approach to a standard MIP implementation in CPLEX 12.2. Their approach was able to close several 50 activity instances, while a normal MIP implementation in CPLEX often failed to solve the instance.

Another extension of the MRIP with generalised temporal constraints and so-called cumulative resources was proposed by Bartels (2009). Cumulative resources are similar to renewable resources but are used to model storage. The start and finish of an activity can change the resource level in a positive or negative way. Bartels proposed two application cases where one considers the dismantling of nuclear power plants (RBP). There, at most two modes are available for each activity but only the resource usage can vary and not the activity processing time. The author used projects with $20-60$ activities and 6 renewable resources in the computational study. The objective is to minimise the net present value associated with mode-dependent costs (which is similar to the nonrenewable resource in the MRIP setting but time dependent). The second area of application models is the scheduling of testing in the automotive industry (VTP). Here, tests (modelled as activities) are assigned to experimental vehicles (represented by cumulative resources) through different modes and the goal is to minimise the total number of utilised experimental vehicles. Again, the processing time does not vary with the mode choice as only resource utilisation is affected by the mode. In Bartels and Zimmermann (2009, 2015), this problem type is further analysed and a priority rule-based schedule generation scheme as well as a GA are utilised. The authors applied these procedures to instances with 20 and 600 activities. Each instance incorporates one renewable resource to model the construction of the experimental vehicles and a cumulative resource for each experimental vehicle.

The problem extension with both a tardiness penalty and multiple modes (MRIPT) was studied by Gerhards and Stürck (2018). They proposed a hybrid large neighbourhood search (LNS) procedure that uses MIP techniques to solve subproblems exactly. They carried out experiments on adapted 30 activity MRCPSP instances of the PSPLIB.

Several other extensions of the resource investment problem exist, where the authors adapted the objective function. In the work of Najafi and Niaki (2006), a RIP extension with discounted cash flows and net present value maximisation is proposed and a GA was applied to the problem. Najafi et al. (2009) extend this setting further by adding generalised precedence constraints to the problem and also applied a GA. A RIP extension with net present value maximisation and tardiness penalty was studied by Najafi and Azimi (2009) and a priority rule-based heuristic was proposed. Yamashita et al. (2007) added uncertainty to the activity durations in the RIP. The authors applied a SS with PR to different scenarios to obtain a robust solution.

In Table 1, we give an overview of the instance properties used in multi-mode RIP studies. Here, $|M|$ is the maximum number of modes per activity. It is clear that most of the studies only address small projects with rather few activities. The majority of the instances are based on MRCPSP instances and only the work of Gerhards and Stürck (2018) used nonrenewable resources (when nonrenewable

Table 1 Instances used in the existing studies of the MRIP

References	Problem	n	$\|M\|$	$\|\mathcal{R}\|$	$\|\mathcal{R}^n\|$
Hsu and Kim (2005)	MRIP	10–30	3	4	0
Yamashita and Morabito (2009)	MRIP	15	3	4	0
Colak and Azizoglu (2014)	MRIP	10–100	10	1	0
Qi et al. (2015)	MRIP	10–30	3	4	0
Coughlan et al. (2015)	MRIP calendars	50	3	2	0
Bartels (2009) (RBP)	MRIP/max with cumulative resources	20, 40, 50, 60	2	6	1
Bartels (2009) (VTP)	MRIP/max with cumulative resources	20, 600	–	1	–
Gerhards and Stürck (2018)	MRIPT	30	3	2	2

resources were available in the original MRCPSP instance, the other authors treated them as renewable resources). Hence, it is desirable to have a more challenging and publicly available benchmark set of instances for future studies.

4 Lower and upper bounds

Next, we present several procedures to obtain lower (Sect. 4.1) and upper bounds (Sect. 4.2) for the MRIP. We apply these methods in Sect. 6 to evaluate the novel set of benchmarks instances.

4.1 Lower bounds

In order to obtain lower bounds, we use four formulae as well as the linear programming relaxations of different mathematical formulations and the destructive improvement approach. Two rather simple lower bounds for the minimum resource level for the renewable resources in the RIP presented by Drexl and Kimms (2001) can be adapted for the multi-mode setting ($\underline{a}_k^1$ and $\underline{a}_k^2$). We calculate how many units for a resource $k \in \mathcal{R}$ are needed so that every activity can be performed in its least resource consuming mode (with $r_{ik}^{\min} = \min_{m \in M_i} r_{imk}$ we denote the minimum resource demand of activity i for resource k). Hence, the minimum resource level for the first method is as follows.

$$\underline{a}_k^1 = \max_{i \in A} \left\{ r_{ik}^{\min} \right\} \qquad k \in \mathcal{R}. \tag{9}$$

As a second way to compute a lower bound on the minimal consumption, we can distribute the required resources equally over the planning horizon. Hence, we divide the sum of the minimal products of the resource consumption and the duration of all activities by the due date D.

$$\underline{a}_k^2 = \frac{\sum_{i \in A} \min_{m \in M_i} \{r_{imk} \cdot d_{im}\}}{D} \qquad k \in \mathcal{R}. \tag{10}$$

Next, we utilise so-called *core times* to compute bounds on the minimal consumption (cf. Klein and Scholl 1999). The core time of an activity i is the periods where the activity has to be processed, i.e. after its latest start LS_i and before it earliest finish $EF_i = ES_i + \min\{d_{im}\}$. So, this concept uses the precedence relations as well as the due date as an upper bound on the project makespan. However, if the due date is large or the minimum activity duration is small, this core time interval can be empty. The bound is computed by checking each period and summing up the minimal resource consumptions of activities that have a core time in this period.

$$\underline{a}_k^3 = \max_{t \in \{0,\ldots,D\}} \sum_{i \in A : LS_i \le t \le EF_i} r_{ik}^{\min} \qquad k \in \mathcal{R}. \tag{11}$$

As a fourth way to compute minimal resource consumptions, we want to identify triplets of activities that cannot be scheduled sequentially, and therefore, at least two of them have to be processed with some overlap. We look at triplets of activities $i, j, l \in A$ with no precedence relations among them. There are six potential orderings of them (i.e. $i \prec j \prec l, i \prec l \prec j, j \prec i \prec l, j \prec l \prec i, l \prec i \prec j$ and $l \prec j \prec i$) and we check whether at least one of them is feasible with respect to the due date. If not, at least two of the three activities have to be carried out in intersecting time intervals and we can add the minimal resource usages. For example, let us assume we investigate the ordering $i \prec j \prec l$, i.e. activity i has to be finished before j can start and j has to be finished before l can start. This means that we can compute a new earliest start time $\overline{ES}_j = \max\{ES_j, ES_i + d_i^{\min}\}$ and latest start time $\overline{LS}_j = \min\{LS_j, LS_l - d_j^{\min}\}$. The ordering is not feasible if $\overline{LS}_j < \overline{ES}_j$. If all six potential orderings are infeasible with respect to the due date, then we add the triplet to the set of infeasible triplets IT. For each triplet in IT, at least two activities have to be scheduled with overlap and we get the following lower bound for the resource usage:

$$\begin{aligned} \underline{a}_k^4 = \max_{(i,j,l) \in IT} \{ \min\{ & \max\{r_{ik}^{\min} + r_{jk}^{\min}, r_{lk}^{\min}\}, \max\{r_{ik}^{\min} + r_{lk}^{\min}, r_{kl}^{\min}\}, \\ & \max\{r_{jk}^{\min} + r_{lk}^{\min}, r_{kl}^{\min}\}\}\} \qquad k \in \mathcal{R}. \end{aligned} \tag{12}$$

For a nonrenewable resource, the only simple lower bound $\underline{R}_k^n$ is the sum of the minimal consumptions of all activities.

$$\underline{a}_k^n = \sum_{i \in A} \min_{m \in M_i} \{r_{imk}^n\} \qquad k \in \mathcal{R}^n. \tag{13}$$

By multiplying these minimal resource levels with the corresponding resource cost factors and summing them, we get the following simple lower bound for the MRIP.

$$LB^0 = \sum_{k \in \mathcal{R}} c_k \cdot \max\left\{\underline{a}_k^1, \underline{a}_k^2, \underline{a}_k^3, \underline{a}_k^4\right\} + \sum_{k \in \mathcal{R}^n} c_k^n \cdot \underline{a}_k^n. \tag{14}$$

We compare LB^0 with bounds obtained by solving the linear programming (LP) relaxation of the MIP. Therefore, we use the mathematical formulation displayed in (1)–(7) (PDT) and we refer to the objective value of this LP relaxation as LB^1. Furthermore, we also use a mathematical formulation where the precedence relations are modelled in a disaggregated way. The constraints are displayed in (15). Here, for each pair $(i,j) \in E$ and for each time period t of the planning horizon a constraint is added that enforces the precedence relations. When the right side of (15) is equal to 1, i.e. the successor j starts before or in period t, then it forces the left side to take a value of 1 as well. Hence, the predecessor i has to be started before $t - d_{im}$ depending on the mode m.

$$\sum_{m \in M_i} \sum_{\tau \leq t - d_{im}} x_{im\tau} \geq \sum_{m \in M_j} \sum_{\tau \leq t} x_{jm\tau} \quad \forall (i,j) \in E, t = 1, \ldots, D \tag{15}$$

Artigues (2017) reported that this disaggregated formulation is stronger (w.r.t to the LP relaxation) than the aggregated formulation displayed in constraints (3) which can be seen since constraints (3) are implied by (2) and (15). The "disaggregated discrete time formulation based on pulse variables" (PDDT) is defined by equations (1), (2), (4)–(7) and (15). With LB^2, we refer to the lower bound obtained by solving the LP relaxation of the PDDT formulation. Note that depending on the value of D, the PDDT formulation can contain considerable more constraints than the PDT formulation, and thus, setting up the mathematical model in the solver most likely takes a longer time.

Another method for computing lower bounds is the *destructive improvement* strategy introduced by Klein and Scholl (1999) for the RCPSP. It is an iterative approach that is started with LB^0 as a starting lower bound $\underline{B}$. In each iteration, we try to proof that if we take $\underline{B}$ as an upper bound on the objective value, then no feasible solution can exist. If we succeed with the proof, we know that $\underline{B} + 1$ is a valid lower bound for the instance and we can use this value in the next iteration. This process is repeated until the proof of infeasibility fails. For the proof, we either use the PDT formulation and a MIP solver (LB^3) or the CP formulation (displayed in "Appendix 1.3") and the IBM ILOG CP Optimizer solver (LB^4). In each iteration, we add an extra constraint that bounds the objective value by the current upper bound $\overline{B}$ to the respective problem formulation. To limit the overall run-time of the procedure, we allow the solver a run-time of 60 s for each iteration. If no infeasibility can be detected after that time, the procedure stops and returns the best-known lower bound. However, if the respective solver finds a feasible solution in an iteration, this solution has to be optimal and the procedure terminates as well.

4.2 Upper bounds

To obtain upper bounds, i.e. the costs of feasible solutions, we use different approaches: On the one hand, we implement heuristic procedures such as a multi-start local search (MLS), a simulated annealing (SA) procedure and an adaptation of the priority rule heuristic (PRH) of Hsu and Kim (2005). On the other hand, we also apply

exact methods such as MIP and CP solvers that are able to detect optimality for some instances (although with a run-time restriction this is not always the case).

Before we explain how the MLS and the SA work in particular, we explain the schedule generation scheme (SGS) that is utilised by both methods. In our case, we use a serial SGS that takes a scheduling sequence $\mathbf{S}$ and a mode $\mathbf{M}$ as an input and tries to build a feasible solution (i.e. a schedule with start and finish times and resource allocations). The SGS that we apply works similar to Algorithm 3.7.1 introduced by Neumann et al. (2003) but also has some slight differences. We schedule the activities one at a time and in the order specified by the sequence $\mathbf{S}$ (in Neumann et al. 2003 the SGS schedules critical activities first and then selects the next activity according to a priority rule; one could interpret our scheduling sequence as a special kind of priority rule). So, the SGS ends after $n + 2$ iterations and we expect the scheduling sequence to respect the precedence constraints. In iteration l of the SGS, it tries to schedule activity $i = \mathbf{S}_l$ in mode $m = \mathbf{M}_i$ at the least cost increasing feasible time period that respects the precedence relations. Thus, the SGS checks all feasible start times $t \in \{ES_i, \ldots, LF_i - d_{im}\}$ and computes the cost increase in the partial schedule with the activity added at each particular start time. If there are multiple start times with the same cost increment, the earliest one of them is assigned. It is possible that the SGS may not find a feasible start time with respect to the due date since the durations of the chosen modes in $\mathbf{M}$ are too high. If the SGS detects such an infeasibility, it fails and does not return a feasible schedule.

First, we explain the concept of the multi-start local search method that we used. Algorithm 1 depicts the outline of the MLS. We initialise the current best mode vector $\mathbf{M}^b$ by choosing the mode with minimal duration for each activity (ties are resolved arbitrarily). This initial mode vector is feasible with respect to the due date D. Next, we determine an initial scheduling sequence $\mathbf{S}^b$ that respects the precedence relations. This means, that a successor j cannot occur at an earlier position in the sequence $\mathbf{S}^b$ than its predecessor i if $(i, j) \in E$. We generate such a sequence by adding the activities one at a time to the sequence. An activity can only be added to the sequence if it is not part of the sequence yet and if all of its predecessors are already in the sequence. If more than one activity is eligible, we either choose one arbitrarily among the eligible activities (random generation) or choose the activity with the highest value of some priority rule. The priority rules that we used in this process are the following: minimum LST, minimum LFT, minimum slack (i.e. $LFT - EFT$), minimum number of successors, maximum number of successors and maximum rank positional weight (cf. Kolisch 1996). We generate a sequence for each priority rule and one using the random selection technique and use the SGS and the mode vector $\mathbf{M}^b$ to translate it into a schedule. We keep the scheduling sequence with the lowest cost value and store it as $\mathbf{S}^b$.

Algorithm 1 Multi-start local search (MLS)

1: Assign an initial mode vector $\mathbf{M}^b$ choosing the minimal duration modes
2: Compute initial best scheduling sequence $\mathbf{S}^b$ with priority rules and random sequence
3: Store best costs c^b after scheduling with $\texttt{SGS}(\mathbf{M}^b, \mathbf{S}^b)$
4: **repeat**
5: $\quad\mathbf{M}^p \leftarrow \texttt{Perturb}(\mathbf{M}^b)$
6: $\quad\mathbf{S}^p \leftarrow \texttt{Perturb}(\mathbf{S}^b)$
7: $\quad(\mathbf{M}^p, \mathbf{S}^p) \leftarrow \texttt{LocalSearch}(\mathbf{M}^p, \mathbf{S}^p)$
8: $\quad$**if** $c(\mathbf{M}^p, \mathbf{S}^p) < c^b$ **then**
9: $\quad\quad c^b \leftarrow c(\mathbf{M}^p, \mathbf{S}^p)$
10: $\quad\quad\mathbf{S}^b \leftarrow \mathbf{S}^p$
11: $\quad\quad\mathbf{M}^b \leftarrow \mathbf{M}^p$
12: $\quad$**end if**
13: **until** time limit is reached

In the main loop of Algorithm 1, we start by assigning a perturbed mode vector, called $\mathbf{M}^p$ in line 5. Here, we select a random number of activities and change their mode from the one stored in $\mathbf{M}^b$ to an arbitrary one. For the remaining activities (that were not selected), we assign in $\mathbf{M}^p$ the same mode as in $\mathbf{M}^b$. The current best sequence $\mathbf{S}^b$ is also modified (by extracting a random number of activities from the sequence and inserting them at a random, precedence feasible position in the modified sequence) and then copied to $\mathbf{S}^p$. We use these two steps to obtain a new start point for the local search. In order to limit the degree of diversification from the current best solution $\mathbf{M}^p$ and $\mathbf{S}^p$, we limit the number of changes by $n \cdot \pi_M^{\max}$ and $n \cdot \pi_S^{\max}$, respectively. Here, $\pi_M^{\max}, \pi_S^{\max} \in [0, 1]$ are parameters of the MLS. Then, we perform a local search starting at $\mathbf{M}^p$ and $\mathbf{S}^p$ in line 7. This local search procedure aims to improve the solution costs by altering $\mathbf{M}^p$ and $\mathbf{S}^p$ locally. First, the LS procedure checks if altering the chosen mode of an activity leads to lower costs by applying the SGS to the altered mode vector and the unchanged sequence. After checking all possible mode changes of an activity, the LS explores all feasible (with respect to precedence relations) swaps in the scheduling sequence. So, it exchanges the activities at two positions of the scheduling sequence and if the sequence still respects the precedence relations, it checks if the application of the SGS leads to a lower cost value. If an alteration of $\mathbf{M}^p$ or $\mathbf{S}^p$ leads to an improvement, we apply the change and repeat the local search procedure with the updated mode vector and scheduling sequence. Hence, we apply a *first improving move* policy. In lines $8-12$, we update the current best mode vector and scheduling sequence if the LS found a strictly better cost value. The MLS repeats until the specified time limit is reached.

We briefly explain the priority rule heuristic of Hsu and Kim (2005): First, we initialise the investment upper bound $IUB = LB^1$. Then, the PRH iteratively tries to find a feasible schedule with costs lower than or equal to IUB. If we cannot find such a schedule, we increase IUB by one at the end of the iteration. To obtain a schedule, we schedule one activity at a time. Therefore, we compute for each activity i that is eligible for scheduling (i.e. all of its predecessors are already scheduled) the mode m^* and the start time t^* that results in the lowest priority value $v(i, m^*, t^*)$. Then, the activity i^* with the worst minimum value is

selected for scheduling. The priority value v is a linear combination of two priority functions v^1 and v^2. The parameter ω controls the portion of the two priority functions.

$$v(j,m,t) = \omega \cdot v^1(j,m,t) + (1-\omega) \cdot v^2(j,m,t) \\ i \in A, m \in M_i, t = ES_i, \ldots, LF_i - d_{im}. \tag{16}$$

Here, v^1 is called the transformed slack priority function and measures how the selected start time and mode influence the set of remaining start times for the successors of this activity (it favours early finish times such that many possible start times remain for the successors). Priority function v^2, called transformed investment priority function, calculates how much it costs to schedule an activity in a specific mode and a start time with respect to the increase in costs of the partial solution (in contrast to the original work, we also consider nonrenewable resources and their costs). If the costs excel the given bound IUB, the v^2 value is ∞. The scheduling procedure stops, if either a start time for each activity was determined or the costs of the (partial) solution excelled the given upper bound IUB. The whole priority rule heuristic stops once a feasible solution is found.

As a second metaheuristic approach, we adapted the simulated annealing procedure with reheating presented by Józefowska et al. (2001) for the MRCPSP to fit our MRIP setting. Here, in each iteration, we generate a solution candidate by perturbing the currently best-known solution. It is accepted if it has a better cost value or with some probability that depends on the delta of the cost values as well as the so-called *temperature*. The temperature is used to control the acceptance rate of worse solutions during the search and decreases constantly by a cooling factor α. However, we also use reheating to escape local optima when the search is stuck. The total number of reheats is one of the algorithm parameters of the SA approach as well as the cooling factor α.

Next, we explain the set-up of the MIP experiments. We implemented six formulations: the two based on pulse variables PDT (displayed in (1)–(7)) and PDDT (displayed in (1), (2), (4)–(7) and (15)). Furthermore, we adapted two formulations based on step variables (SDT and SDDT) and two based on on/off variables (OODT and OODDT) which are displayed in more detail in "Appendix 1.1" and 1.2, respectively. In order to model and solve the MIP, we used Gurobi Optimizer 9.0 via the C# API. For large instances, setting up the model can require some time (in our experiments, the maximal measured set-up time was 8.16 s), while for the smaller instances, no big difference in set-up times between the two formulations was measured.

Another exact approach that becomes more and more relevant in the field of project scheduling is constraint programming. Constraint programming solvers became more efficient in recent years and they are applied to a growing number of scheduling problems (e.g. Kreter et al. 2018 and Schnell and Hartl 2016). The CP formulation of the MRIP used in this study is displayed in "Appendix 1.3". We implemented the CP model displayed in (47)–(54).

5 Instance generation and benchmark library

In this section, we explain how we generated the benchmark instances for the MRIP. Our main concerns with the instances are diversity of the instances with respect to certain characteristics and that the feasible mode space cannot be reduced easily. We used the following characteristics to generate different instances: number of activities n, maximum number of modes per activity $|M|$, number of renewable resources $|R|$, due date factor θ, order strength OS and resource factor RF. Here, the parameter order strength is a measure of the number of precedence relations (Mastor 1970). It is defined as the fraction of precedence relations in E compared to the total number of possible relations and, hence, is an indicator whether the precedence structure of the project is more parallel or more serial (Schwindt 1998 showed that OS is a good approximation for the restrictiveness of the precedence relations). The resource factor value is the average portion of different resources required for the processing of an activity in a mode (Kolisch and Sprecher 1997). Note that it only varies for the renewable resources. For the single nonrenewable resource, $RF = 1$ for all instances since each mode has a positive resource requirement for the nonrenewable resource except the modes of the dummy start and finish activities. The due date factor θ is used to compute the due date of the project based on the earliest start time ES_{n+1} of the dummy end activity (calculated by forward calculation). We calculate the due date as follows:

$$D = \texttt{round}(\theta \cdot ES_{n+1}). \tag{17}$$

The `round` function in equation (17) applies the *rounding to the nearest even number* strategy in the case of midpoint values. Table 2 displays the values that we used. For each parameter combination, we generated 5 instances, giving us in total 4950 instances. As shown in Sect. 2, it is sufficient to consider only one nonrenewable resource.

Concerning the mode space, the instances should neither contain *inefficient* nor *infeasible* modes. We call a mode $m \in M_i$ of activity i *inefficient* if there is another mode $m' \in M_i$ for that activity such that the other mode has a shorter or equal duration and it has lower or equal resource requirements for all resources (cf. Kolisch et al. 1995). It would never be beneficial to include an *inefficient* mode m into a solution since with m' the resource usage would be lower, and hence, we can omit mode m from the mode space. It is possible to check if a mode is *inefficient* in polynomial

Table 2 Parameter values of the new instances

Parameter	Values
n	$\{30, 50, 100\}$
$\|M\|$	$\{3, 6\}$
$\|\mathcal{R}\|$	$\{2, 4, 8\}$
θ	$\{1.2, 1.4, 1.6, 1.8, 2\}$
OS	$\{0.25, 0.5, 0.75\}$
RF	$\{0.25, 0.5, 0.75, 1\}$

time. We call a mode $m \in M_i$ *infeasible* if its duration d_{im} is too long to finish the project before the due date. In mathematical terms, this happens if the difference between the latest finish time (LF) of the activity and the earliest start time (ES) is smaller than the duration, i.e.:

$$LF_i - ES_i < d_{im}. \tag{18}$$

Especially when using project data that was generated for the MRCPSP and when the due date is very close to ES_{n+1}, many modes are *infeasible* and can be omitted from the mode space. Again, we can check in polynomial time if a mode is *infeasible* since we can calculate the *ES* and the *LF* in polynomial time with the FBC.

Gerhards and Stürck (2018) adapted MRCPSP instances from the PSPLIB with 30 activities per project. They also computed the due date as in (17) and used values of $\theta \in \{1.0, 1.1, \ldots, 1.5\}$. However, they did not investigate if the duration of each mode is short enough for the resulting due dates. In Table 3, we present the average number of *infeasible* modes for the different values of θ as well as the maximum and minimum proportion of *infeasible* modes for single instances (labelled min and max in the table). Especially for $\theta = 1.0$ and $\theta = 1.1$, many modes are *infeasible*. Furthermore, we also computed how many modes become *inefficient* when we try to make the *infeasible* modes feasible. We altered the duration of all *infeasible* modes by setting $d_{im} = LF_i - ES_i$. Again, for low values of θ many modes can be omitted from the mode space since they are *inefficient* (the altered modes have a shorter duration and dominate unchanged ones). For this reason, we consider in this study slightly larger values for the parameter θ and check during the generation of the instances for *infeasible* and *inefficient* modes.

In the following, we describe how we computed each instance with the desired properties. First, we used the network generator *RanGen* Demeulemeester et al. (2003) to generate an activity-on-the-node network with the specified number of activities and *OS* value (this gives us the set E of precedence relations). For each activity $i \in A$, the cardinality of the mode set is set to the desired number, i.e. $|M_i| = |M|$ (except for the dummy activities which have only one mode). We draw for each activity i and all its modes $m \in M_i$ the duration d_{im} as a discrete uniformly distributed random number $\mathcal{U}\{1, 10\}$. Similarly, the resource requirements r_{imk} (r^n_{imk}) for each resource $k \in \mathcal{R}$ ($k \in \mathcal{R}^n$) are also taken from $\mathcal{U}\{1, 10\}$. If the value of $RF < 1$, then we arbitrarily set sufficiently many of the renewable resource

Table 3 *Infeasible* and *inefficient* modes in the adapted PSPLIB instances used by Gerhards and Stürck (2018)

θ	∅ Infeasible (%)	Min (%)	Max (%)	∅ Inefficient (%)	Min (%)	Max (%)
1.0	28.17	11.11	52.22	13.16	1.11	37.78
1.1	12.02	0.00	35.56	2.01	0.00	14.44
1.2	3.54	0.00	22.22	0.39	0.00	5.56
1.3	0.52	0.00	11.11	0.04	0.00	2.22
1.4	0.03	0.00	3.33	0.00	0.00	0.00
1.5	0.00	0.00	0.00	0.00	0.00	0.00

requirements of each mode to 0 such that the resource factor is achieved. After we determined all the resource requirements and the durations of an activity, we check if there are *inefficient* modes. If an *inefficient* mode occurs, we take new random values for the modes that cause the inefficiency. This is repeated until each activity has no *inefficient* modes. Then, we use the forward calculation to compute *ES* and determine the due date *D* as in (17). Based on the due date, we can compute latest finish times *LF* for all activities. Next, we check each activity for *infeasible* modes. If we encounter an *infeasible* mode with some activity, we start over and redraw all values (durations and resource requirements) of all activity modes of the instance. By taking new random values for all of the modes and not just the infeasible ones, we want to avoid that infeasible modes are set to a similar duration (for example, setting the duration to $d_{im} = LF_i - ES_i$ would repair the infeasibility, but could also result in inefficient modes and biased duration values). We repeat this until all modes of all activities are feasible and none of them is dominated. Finally, we determine the random cost factors c_k for all renewable resources $k \in \mathcal{R}$ from $\mathcal{U}\{1, 10\}$. Although, we only investigate the MRIP without tardiness permitted in this study, we also added a value for the tardiness cost factor (also from $\mathcal{U}\{1, 10\}$) that can be useful in future work. In the next section, we will investigate the novel instances.

For most instances (3725 of 4950), we drew feasible mode durations on the first try. However, on average we had to redraw 385 times per instance until all modes were feasible. The maximum number of redraws was 67,653 for instance MRIP_30_79. We analysed if the redrawing has an effect on the distribution of the mode duration values. Therefore, we performed a Pearson's chi-squared test of goodness of fit (Cochran 1952) with significance level $\alpha = 0.01$ to check whether the duration data fits to a discrete uniform distribution. Only for 47 of the 4950 instances, we were able to reject this hypothesis, and hence, we suspect that there is no strong bias in the duration values for most of the generated instances.

6 Computational study

To test the "hardness" of the instances, we compute both lower and upper bounds with the procedures presented in Sect. 4 and compare them. To compute upper bounds, we apply metaheuristic search procedures as well as different MIP and CP implementations. The goal is to examine how hard the proposed instances are for these general purpose methods. For the lower bounds, we propose some rather simple approaches and compare them to linear programming relaxations of different mathematical formulations as well as so-called destructive improvement methods. We want to investigate if drawing the cost factors from a uniform distribution has an effect on the "hardness" of the instances. Therefore, we perform experiments with the random cost factor instances and additionally with the same instances, but with all cost factors set to 1 (called equal resource cost factors). We conducted all following computational experiments on an Intel Xeon Silver 4214 CPU running at 2.20 GHz and all procedures were implemented in C#. We utilised Gurobi Optimizer 9.0 to solve LP relaxations and the MIPs. For the CP-based destructive improvement

procedure and the standalone CP implementation, we used IBM ILOG CP Optimizer 12.9.0 (cf. Laborie et al. 2018).

The five lower bound procedures proposed in Sect. 4.1 are applied on the new benchmark instances. In Table 4, we depict the average of the relative improvement over the worst lower bound $LB^{\min}$ for all five lower bounds, i.e. $\frac{LB^i - LB^{\min}}{LB^{\min}}$, $i = 0, \ldots, 4$. If the value of ∅ improvement is close to 0%, then the respective lower bound value was close to the minimum value among the calculated bounds. Hence, the higher the ∅ improvement the better. We used random resource cost factors for the results depicted in Table 4. For instances with unit resource cost factors, we get a similar trend.

Clearly, LB^0 (the bound based on minimum resource level formulae) is always the worst lower bound. Only for instances with a small number of modes ($|M| = 3$) or a high resource factor ($RF = 1$), the gap between the more advanced procedures gets smaller. The LP relaxation-based lower bound with disaggregated precedence constraints LB^2 performs best on average. Since the disaggregated formulation is stronger than the aggregated one (cf. Artigues 2017), LB^1 (based on the aggregated precedence constraints) performs slightly worse than LB^2 regarding the relative improvement. The destructive improvement methods (LB^3 and LB^4) work

Table 4 Average improvement (with random resource cost factors) for the different instance parameters

		∅ Improvement				
		LB^0 (%)	LB^1 (%)	LB^2 (%)	LB^3 (%)	LB^4 (%)
All		0.00	119.46	122.99	119.57	119.82
n	30	0.00	122.02	126.78	150.52	128.27
	50	0.00	117.07	120.56	124.11	114.65
	100	0.00	119.28	121.63	84.08	116.54
$\|\mathcal{R}\|$	2	0.00	104.03	106.48	99.17	102.12
	4	0.00	112.97	116.16	112.16	114.83
	8	0.00	137.51	142.19	142.28	138.09
$\|M\|$	3	0.00	58.90	60.68	61.47	63.42
	6	0.00	180.01	185.29	177.66	176.22
OS	0.25	0.00	143.38	145.80	145.35	142.15
	0.5	0.00	119.64	123.27	119.19	118.83
	0.75	0.00	95.35	99.90	94.16	98.47
θ	1.2	0.00	130.15	135.03	147.78	128.52
	1.4	0.00	128.70	133.05	139.47	127.60
	1.6	0.00	121.50	125.09	121.74	121.55
	1.8	0.00	112.76	115.37	102.91	114.43
	2	0.00	104.19	106.40	85.93	107.00
RF	0.25	0.00	146.83	154.11	168.96	155.25
	0.5	0.00	158.22	163.39	157.84	157.53
	0.75	0.00	121.43	123.61	110.42	119.11
	1	0.00	60.48	61.21	57.52	59.20

especially well on small- to medium-sized instances but are outperformed by the LP-based approaches on the $n = 100$ instances. However, LB^3 finds optimal solution for 31 instances, while LB^4 can solve 4 instances to optimality. In absolute terms, LB^3 gives the best lower bound for 3736 instances (and 3403 of those bounds were solely found by this procedure). Yet, the performance of the MIP solver seems to be significantly worse for the instances with 100 activities, as we can see a drop in the average improvement there.

Furthermore, in Table 5, we display the average computation time (in s). As expected, LB^0 has a significantly lower running time than the other procedures. The PDT LP relaxation (LB^1) is solved about 10 times faster than the disaggregated version. While the MIP-based destructive improvement procedure takes the longest computation time, it also closes most instances and performs best on the instances with 30 and 50 activities. The CP-based variant is not able to find as many optimal solutions as its MIP-based counterpart, yet, it terminates faster. For instances with 30 and 50 activities, LB^2 seems to have the best quality per computation time ratio, while on the larger 100 activities instances, LB^1 seems most efficient. However, we solved the LP relaxations (LB^1 and LB^2) much faster when the mode count was higher. To investigate this unexpected behaviour, we modified the instances

Table 5 Average computation time in seconds (with random resource cost factors) for the different instance parameters

		∅ Computation time				
		LB^0	LB^1	LB^2	LB^3	LB^4
All		0.05	12.47	145.39	2491.07	185.71
n	30	0.02	0.52	2.13	711.89	241.64
	50	0.02	2.50	17.30	1748.01	158.95
	100	0.10	34.40	416.73	5013.31	156.56
$\lvert\mathcal{R}\rvert$	2	0.03	1.25	64.60	1503.81	97.68
	4	0.04	6.12	130.04	2001.14	165.81
	8	0.07	27.25	221.31	3721.45	271.65
$\lvert M\rvert$	3	0.05	18.44	234.38	2100.45	195.22
	6	0.05	6.50	56.39	2881.69	176.21
OS	0.25	0.04	1.71	20.13	2610.90	120.65
	0.5	0.05	7.67	89.94	2863.30	152.24
	0.75	0.05	28.05	326.08	1999.01	284.26
θ	1.2	0.05	1.39	11.96	2290.56	224.11
	1.4	0.05	2.79	30.14	2783.63	188.63
	1.6	0.05	6.27	81.81	2674.72	172.52
	1.8	0.04	13.83	200.08	2470.73	172.57
	2	0.04	38.09	402.93	2235.71	170.74
RF	0.25	0.05	4.44	123.20	1721.90	321.36
	0.5	0.04	8.65	151.83	1947.00	213.10
	0.75	0.04	15.60	122.44	2813.10	143.85
	1	0.04	18.53	176.67	3225.89	109.77

with three modes per activity by duplicating each mode of each activity and solving the modified instance again as an LP relaxation. Note that the objective value does not change when solving this modified instance. However, the average computation time decreased from 18.44 to 15.96 s for the aggregated variant and from 234.38 to 163.49 s for the disaggregated version. So, basically the same instance was solved up to 30% faster just by adding the identical modes. Why the LP solver is so much faster with these additional decision variables is still an open problem and further research is needed here.

Next, we analyse the upper bound procedures presented in Sect. 4.2. Therefore, let us first examine how many instances the MIP and CP procedures solved to optimality within a time limit of one hour. When we used equal resource cost factors, the PDT formulation MIP solved 17.4% of the instances to optimality, whereas the CP solver found optimal solutions for 30.4% of the instances. Using random cost factors, PDT solved only 9.3% and CP 28.4% of the instances to optimality. In total, the exact procedures solved 1430 instances to optimality, and hence, we found optimal solutions for less than 29% of the instances with random cost factors. The random cost factors make the instances more challenging, especially for the MIP-based approaches. Therefore, we take a closer look at these instances.

In Table 6, we can observe how the exact approaches performed given different run-time limits (60, 600, and 3600 s). Surprisingly, the CP implementation finds more optimal solutions in 60 s than any of the MIP approaches in one hour. The aggregated versions of the step-based and the on/off-based formulations cannot find feasible solutions for all of the instances, even with a time limit of one hour. The disaggregated variants find at least one feasible solution for each instance except for PDDT and OODDT with the 60 second time limit. PDT can solve the most instances to optimality among the MIP-based approaches. However, the CP approach seems to be much better at proving optimality of a solution and, hence, achieves also faster average run-times.

Table 6 Percentage of feasible and optimal solutions and average computation times in seconds for the instances with random cost factors

Max time	∅ Feasible			∅ Opt			∅ Time		
	60 (%)	600 (%)	3600 (%)	6 (%)0	600 (%)	3600 (%)	60	600	3600
PRH	100.0			0.0			12.5		
MLS	100.0	100.0	100.0	0.0	0.0	0.0	60.0	600.0	3600.0
SA	100.0	100.0	100.0	0.0	0.0	0.0	60.0	600.0	3600.0
CP	100.0	100.0	100.0	14.6	22.6	28.4	53.5	485.7	2688.6
PDT	100.0	100.0	100.0	0.9	4.3	9.3	60.1	584.1	3361.3
PDDT	97.8	100.0	100.0	0.8	4.2	8.8	60.8	585.9	3372.6
OODT	36.0	62.7	81.4	0.1	1.4	4.2	61.1	597.0	3503.4
OODDT	94.1	100.0	100.0	0.2	3.0	8.3	61.1	591.5	3406.8
SDT	93.7	97.5	99.4	0.0	0.7	2.5	60.4	598.4	3547.8
SDDT	100.0	100.0	100.0	0.2	1.8	5.3	60.4	594.4	3477.8

In Table 7, we can observe how many instances were solved optimally by CP or one of the MIP formulations after one hour of computation depending on the respective instance parameter. We see that for "small" instances, i.e. the instances with only 30 activities, CP outperforms all other approaches and is the only procedure that finds optimal solutions for instances with 100 activities. In general, instances with 30 activities, a small $RF = 0.25$ or a small due date factor are more likely to be solved to optimality by one of the approaches. A similar phenomenon as with the LP relaxations occurs with the PDDT MIP approach where more optimal solutions are found for instances with higher mode count as the MIP solver has to solve several LPs during the search.

Next, we compare the results of the exact procedures to the MLS, the SA approach and the PRH. To calibrate the algorithm parameters of the MLS, PRH and SA, we used the software package `irace` López-Ibáñez et al. (2016) and a set of 990 training instances (with the same instance parameters). After performing 1000 experiments with each algorithm, the best parameter values on the training instance set are $\pi_S^{\max} = 0.92$, $\pi_M^{\max} = 0.2$ and $\omega = 0.28$ for the MLS and the PRH. For the SA approach, the number of reheats was 23, 880 and 5280 and the temperature cooling factor α was 0.52, 0.9 and 0.9 for a run-time limit of 60, 600 and 3600 s, respectively.

Table 7 Percentage of optimal solutions for different instance parameters after 3600 s of run-time (for the instances with random cost factors)

		CP (%)	PDT (%)	PDDT (%)	OODT (%)	OODDT (%)	SDT (%)	SDDT (%)
All		28.4	9.3	8.8	4.2	8.3	2.5	5.3
n	30	62.7	25.2	23.8	12.2	22.8	7.3	14.6
	50	22.0	2.6	2.6	0.4	2.1	0.1	1.3
	100	0.6	0.0	0.0	0.0	0.0	0.0	0.0
$\lvert\mathcal{R}\rvert$	2	27.5	10.7	11.3	3.5	9.9	3.5	8.1
	4	32.4	10.6	10.0	5.8	9.4	2.9	5.7
	8	25.2	6.8	5.7	3.2	6.1	1.3	2.8
$\lvert M\rvert$	3	34.1	9.7	8.6	5.7	10.1	3.4	6.6
	6	22.8	8.8	9.0	2.8	6.6	1.5	4.0
OS	0.25	21.8	9.1	8.2	5.1	7.9	2.7	4.7
	0.5	28.1	8.4	7.6	3.6	7.6	2.2	5.0
	0.75	35.4	10.2	10.6	3.9	9.4	2.6	6.1
θ	1.2	34.8	15.3	15.4	8.3	13.7	6.6	10.9
	1.4	30.2	11.1	11.4	4.6	10.0	3.2	7.9
	1.6	26.6	7.7	7.7	3.5	7.7	1.5	3.9
	1.8	25.4	6.7	5.5	2.2	5.7	0.6	2.1
	2	25.4	5.6	4.0	2.4	4.4	0.5	1.6
RF	0.25	62.8	28.7	25.0	16.3	26.4	7.0	14.1
	0.5	37.9	11.2	11.5	3.6	9.6	3.3	7.2
	0.75	17.7	2.9	3.1	0.8	2.6	1.0	2.2
	1	6.8	0.7	1.0	0.1	0.6	0.2	0.7

We see that using mostly the transformed investment priority function performs better for PRH (in Hsu and Kim 2005, an ω value of 0.4 performed best). On average, the PRH took 12.5 s to compute a feasible solution. The fastest PRH running time took 0.08 s, while the maximum computation time in our experiments was 258.5 s for one instance.

Metaheuristic procedures are in general not able to detect if a solution is optimal or not. So, we compare the procedures based on the average relative deviation (∅ RD, cf. (19)) from the respective best-known solution value ($UB^{\min}$) found by the four procedures. So, if this average is 0%, this means that the procedure always found a solution with lowest objective function value for each instance.

$$RD = \frac{UB - UB^{\min}}{UB^{\min}}. \tag{19}$$

We also compare them by the average relative deviation from the best-known lower bound ($\varnothing RDLB$, cf. (20)).

$$RDLB = \frac{UB - LB^{\max}}{LB^{\max}}. \tag{20}$$

Tables 8 and 9 depict the average relative deviations $\varnothing RD$ and $\varnothing RDLB$ of the tested procedures, respectively. Note, that SDT and OODT did not find a feasible solution for each instance and we omitted instances with no solution in the calculation of the average for these two formulations. This means that the true average for those procedures may be higher than displayed and a comparison with the other methods is difficult. We observe that the PDT formulation reaches the lowest average deviation over all instances although CP was able to find more optimal solutions. One could suspect that the stronger PDDT formulation should also have a better performance than the weaker aggregated PDT variant. But we need to keep in mind, that solving the disaggregated LPs takes longer and the extra constraints also need more memory, and hence, it is not a priori clear which MIP formulation performs better (cf. Artigues 2017). However, for the project instances with 30 activities, the CP implementation achieves the best results. For instances with more activities, again PDT achieves the best results and also the gap to the metaheuristic procedures grows. Among the heuristic procedures, MLS works best for small- to medium-sized project instances but SA performs better on the 100 activity instances. However, the heuristic approaches are not competitive with the exact methods and there is a lot of potential for improvement.

In Table 9, we see that PDDT, SDDT and OODDT have a lower deviation for instances with a higher number of modes. We think that this phenomenon is related to the one experienced with the lower bounds. When solving the MIP, also several LPs are solved, and hence, we think this is related to the better performance of the disaggregated MIP procedures on instances with 6 modes per activity. A low resource factor of 0.25 seems to make it easier for the CP and PDT formulation which is in line with the findings in Table 7 where the most optimally solved instances also had a $RF = 0.25$. The choice of the due date factor θ has no strong impact on the solution quality of the CP solver. The MIP approaches, especially SDDT and OODDT, perform worse for

Table 8 Average relative deviation (Ø RD) from the best-known solution comparison after 3600 s of run-time for the instances with random cost factors

		PRH (%)	SA (%)	MLS (%)	CP (%)	PDT (%)	PDDT (%)	OODT* (%)	OODDT (%)	SDT* (%)	SDDT (%)
All		31.3	24.3	16.4	2.5	1.4	2.2	2.1	10.7	3.5	8.4
n	30	30.9	23.5	7.2	0.4	1.2	1.1	1.7	1.0	2.1	1.8
	50	31.7	23.7	11.8	1.7	1.6	1.7	2.7	2.0	2.7	2.9
	100	31.4	25.7	30.2	5.3	1.3	3.8	2.0	29.0	5.7	20.6
$\lvert\mathcal{R}\rvert$	2	32.6	19.3	14.4	2.1	0.8	0.8	1.4	6.8	1.2	1.7
	4	31.3	23.9	16.2	2.8	1.3	2.2	2.2	10.9	2.5	4.3
	8	30.4	28.4	18.1	2.5	1.9	3.1	2.8	13.4	6.2	17.5
$\lvert M\rvert$	3	27.9	25.9	13.8	1.8	1.6	3.1	2.4	11.5	3.8	9.7
	6	34.8	22.7	18.9	3.2	1.1	1.3	1.9	9.8	3.1	7.1
OS	0.25	31.8	28.6	19.2	3.7	1.0	1.0	1.2	1.5	1.6	3.4
	0.5	29.0	24.2	17.1	2.6	1.2	1.5	2.0	10.0	3.1	8.4
	0.75	33.2	20.0	12.8	1.2	1.8	4.0	3.8	20.5	5.7	13.5
θ	1.2	30.9	25.0	15.7	2.4	0.9	0.8	1.5	0.7	1.4	1.1
	1.4	32.0	25.7	16.2	2.5	1.1	1.0	1.6	2.2	1.8	1.4
	1.6	31.9	24.4	16.4	2.5	1.4	1.6	2.2	8.1	2.7	6.5
	1.8	31.1	23.7	16.8	2.5	1.6	2.7	2.7	18.2	4.6	12.9
	2	30.7	22.7	16.8	2.5	1.9	4.8	3.0	24.1	7.0	20.2
RF	0.25	28.1	24.5	15.5	1.1	1.1	2.0	2.5	13.2	4.9	8.1
	0.5	32.3	23.4	16.3	2.4	1.3	2.0	2.4	10.2	3.7	7.2
	0.75	32.4	24.8	16.9	3.2	1.5	1.9	2.2	10.2	3.1	8.6
	1	31.5	24.5	16.5	2.8	1.5	2.7	1.6	9.9	2.6	9.7

Table 9 Average relative deviation (∅ RDLB) from the best-known lower bound comparison after 3600 s of run-time for the instances with random cost factors

		PRH (%)	SA (%)	MLS (%)	CP (%)	PDT (%)	PDDT (%)	OODT* (%)	OODDT (%)	SDT* (%)	SDDT (%)
All		50.6	42.5	33.3	17.5	16.3	17.3	15.8	27.7	18.9	25.2
n	30	50.7	42.0	23.2	15.4	16.4	16.3	16.9	16.1	17.4	17.1
	50	53.1	43.7	29.8	18.1	18.1	18.2	17.9	18.7	19.5	19.7
	100	48.1	41.8	47.0	18.8	14.4	17.6	10.1	48.1	19.8	38.8
$\lvert\mathcal{R}\rvert$	2	39.7	25.5	20.3	7.4	6.1	6.2	6.5	12.6	6.5	7.1
	4	47.3	38.7	30.0	15.1	13.6	14.7	13.8	24.7	14.9	17.1
	8	62.2	59.1	46.4	27.3	26.7	28.4	26.0	41.9	32.3	46.9
$\lvert M\rvert$	3	45.8	43.6	29.8	16.1	16.0	17.8	15.4	27.9	18.7	26.2
	6	55.5	41.4	36.9	18.9	16.7	16.9	16.1	27.4	19.0	24.3
OS	0.25	45.0	41.6	30.9	14.0	11.1	11.0	11.0	11.6	11.7	13.9
	0.5	47.3	42.1	33.8	17.2	15.7	16.0	15.9	26.3	18.1	24.6
	0.75	59.6	43.8	35.3	21.2	22.1	24.9	22.6	45.0	27.0	37.2
θ	1.2	45.5	38.8	28.5	13.7	12.1	12.0	12.4	11.9	12.7	12.3
	1.4	49.4	42.1	31.2	15.9	14.4	14.3	14.3	15.7	15.2	14.7
	1.6	51.9	43.1	33.8	17.9	16.7	17.0	16.4	25.0	18.3	23.2
	1.8	52.7	44.0	35.8	19.2	18.3	19.7	18.0	38.6	22.0	32.6
	2	53.8	44.4	37.3	20.5	20.1	23.7	19.1	47.2	26.4	43.2
RF	0.25	49.0	44.6	34.4	17.5	17.7	18.8	16.1	33.0	22.4	26.7
	0.5	53.0	42.7	34.3	18.3	17.2	18.1	17.0	28.2	20.2	24.9
	0.75	52.5	43.8	34.4	18.8	16.9	17.6	16.8	27.5	18.9	26.0
	1	47.6	39.6	30.5	15.2	13.9	15.4	13.3	23.6	15.1	23.7

higher due date factors. OODDT performs best over all procedures on the $\theta = 1.2$ and worst on the $\theta = 2$ instances. In total, most best-known solutions (BKS) were found by CP (2228) followed by OODDT (1362), PDDT (1268) and PDT (1239). The MLS found only 15 BKS and the PRH only a single one, while the SA approach did not find any BKS at all.

7 Summary and conclusions

In this paper, we investigated the multi-mode resource investment problem. It is a prominent project scheduling problem where each activity is processed in one of multiple modes that vary in the activities' duration and resource consumption. We have shown that it is sufficient to consider a single nonrenewable resource with a transformation procedure.

Furthermore, we propose a novel set of benchmark instances for this problem as no common set of instances was used and known in the literature so far. In total, we computed 4950 instances with a diverse set of instance characteristics. We especially assured that none of the modes can be reduced because of infeasibility or inefficiency reasons. By maintaining the website https://riplib.hsu-hh.de, we make these instances available to the public. In addition, researchers can access best-known solution values for the instances as well as share and validate their results on this website. We encourage researchers to test their solution procedures on the benchmark dataset and compare their results.

In extensive computational experiments, we examined lower and upper bounds for the MRIP. For the lower bounds, our experiments revealed that using the LP relaxation of the so-called disaggregated discrete time indexed formulation yields better lower bounds for most instances at hand. However, we also proposed destructive improvement methods that yielded good results for the small- and medium-sized instances and even provided optimal solutions in some cases.

We also tested several procedures to obtain good upper bounds for the MRIP. The metaheuristic procedures, a multi-start local search, simulated annealing, and a priority rule heuristic from the literature, were not able to compete with the MIP and CP implementations. In total, the exact procedures were able to prove the optimality of 1340 of the 4950 instances. That means that over 60% of the instances are still open and we encourage researchers to investigate them further. Our experiments also indicated that the instances are more challenging when we use random cost factors.

For future research, we advise the application of more advanced metaheuristic procedures. In addition, further extensions such as general temporal constraints or the tardiness penalty in the objective function could be an interesting addition to take some important aspects of project scheduling into account.

Acknowledgements Open Access funding provided by Projekt DEAL.

Appendix 1: Additional mathematical model formulations

Here, we present additional mathematical representations of the MRIP. In 1.1 and 1.2, mixed-integer programming models based on different decision variables are presented. They are based on RCPSP models found in Artigues (2017). In 1.3, we display a constraint programming formulation of the MRIP.

Appendix 1.1: Step variables discrete time

First, let us present the formulation based on so-called *step* variables. Here, we have a binary step variable z_{imt} that is 1 if and only if activity i starts in mode m at time t or before. Hence, we can express the start time S_i of an activity by looking at the difference of step variables, i.e.:

$$S_i = \sum_{m \in M_i} \sum_{t=1}^{D} t \cdot (z_{imt} - z_{i,m,t-1}). \tag{21}$$

Next, we show a model using these step variables and the aggregated precedence constraints in (22)–(31)

$$\min \sum_{k \in \mathcal{R}} c_k \cdot a_k + \sum_{k \in \mathcal{R}^n} c_k^n \cdot a_k^n \tag{22}$$

$$s.t. \sum_{m \in M_i} \left(z_{i,m,0} + \sum_{t=1}^{D} (z_{imt} - z_{i,m,t-1}) \right) = 1 \quad \forall i \in A \tag{23}$$

$$z_{imt} \geq z_{i,m,t-1} \qquad \forall i \in A, \forall m \in M_i \tag{24}$$

$$z_{imt} = 0 \qquad \forall i \in A, \forall m \in M_i, \forall t < ES_i \tag{25}$$

$$\sum_{m \in M_i} z_{i,m,LF_i - d_{im}} = 1 \qquad \forall i \in A \tag{26}$$

$$\sum_{m\in M_i}\left(d_{im}\cdot z_{i,m,0}+\sum_{t=1}^{D}(t+d_{im})\cdot(z_{imt}-z_{i,m,t-1})\right)\leq \sum_{m\in M_j}\sum_{t=1}^{D}t\cdot(z_{jmt}-z_{j,m,t-1}) \qquad \forall(i,j)\in E \tag{27}$$

$$\sum_{m\in M_i} r^n_{imk}\cdot\left(z_{i,m,0}+\sum_{t=1}^{D}(z_{imt}-z_{i,m,t-1})\right)\leq a^n_k \quad \forall k\in\mathcal{R}^n \tag{28}$$

$$\sum_{i\in A}\sum_{m\in M_i}\sum_{q=\max(ES_i,t-d_{im}+1)}^{\min(t,LF_i-d_{im})} r_{imk}\cdot(z_{imq}-z_{i,m,q-1})\leq a_k \quad \forall k\in\mathcal{R}, t=0,\ldots,D \tag{29}$$

$$a_k\geq 0 \quad \forall k\in\mathcal{R} \qquad a^n_k\geq 0 \quad \forall k\in\mathcal{R}^n \tag{30}$$

$$z_{imt}\in\{0,1\} \quad \forall i\in A, \forall m\in M_i, t=0,\ldots,D. \tag{31}$$

This formulation is called the aggregated version of the discrete time formulation based on step variables (SDT). The objective function (22) is the same as in the PDT formulation and minimises the resource costs. The equation in (23) ensures that exactly one mode and one step (i.e. a start) is chosen for each activity. The inequalities in (24) let the step variables grow, while the equality in (26) makes sure that the start time of each activity is scheduled before the mode specific latest start time $LF_i - d_{im}$. In (27), we display the aggregated version of the precedence constraints. The next inequalities (28) model the nonrenewable resources. (29) displays the renewable resource constraints. Finally, in (30) and (31), the domains of the decision variables are defined. There is also a version with disaggregated precedence constraints which are displayed in (32).

$$\sum_{m\in M_i} z_{i,m,t-d_{im}}\geq\sum_{m\in M_j} z_{jmt} \qquad \forall(i,j)\in E, t=1,\ldots,D. \tag{32}$$

We call the formulation using constraints (22)–(26), (28)–(31) and (32) the disaggregated version of the discrete time formulation based on step variables (SDDT).

Appendix 1.2: On/off variables discrete dime

The third model utilises so-called on/off variables y_{imt} that are set to 1 if and only if activity i is in process in mode m at time t. An activity i is started at time t or before in mode m if it was in process in at least one of the following times: $t, t-d_{im}, t-2d_{im}, t-3d_{im}, \ldots$. Hence, we get the following relation between the step and the on/off variables:

$$z_{imt} = \sum_{k=0}^{\lfloor t/d_{im} \rfloor} y_{i,m,(t-kd_{im})}. \tag{33}$$

By combining (21) and (33), we get the following formula for the start time of activity i

$$S_i = \sum_{t=1}^{D} t \cdot \left(\sum_{k=0}^{\lfloor t/d_{im} \rfloor} y_{i,m,(t-kd_{im})} - \sum_{k=0}^{\lfloor (t-1)/d_{im} \rfloor} y_{i,m,(t-1-kd_{im})} \right). \tag{34}$$

The model using on/off variables and aggregated precedence constraints (OODT) is displayed in (35)–(45). Since we are in a multi-mode setting, we need to introduce an additional binary variable e_{im} to represent the mode choice. This decision variable is equal to 1 if and only if activity i is processed in mode m.

$$\min \sum_{k \in \mathcal{R}} c_k \cdot a_k + \sum_{k \in \mathcal{R}^n} c_k^n \cdot a_k^n \tag{35}$$

$$s.t. \sum_{m \in M_i} \sum_{k=0}^{\lfloor LF_i/d_{im} \rfloor} y_{i,m,(t-kd_{im})} = e_{im} \quad \forall i \in A \tag{36}$$

$$\sum_{m \in M_i} e_{im} = 1 \qquad \forall i \in A \tag{37}$$

$$\sum_{t=1}^{D} y_{imt} = d_{im} \cdot e_{im} \qquad \forall i \in A, \forall m \in M_i \tag{38}$$

$$\sum_{k=0}^{\lfloor t/d_{im} \rfloor} y_{i,m,(t-kd_{im})} \geq \sum_{k=0}^{\lfloor (t-1)/d_{im} \rfloor} y_{i,m,(t-1-kd_{im})} \quad \forall i \in A, \forall m \in M_i, t = 1, \ldots, D \tag{39}$$

$$\begin{aligned} &\sum_{m \in M_i} \sum_{t=1}^{D} (t + d_{im}) \cdot \left(\sum_{k=0}^{\lfloor t/d_{im} \rfloor} y_{i,m,(t-kd_{im})} - \sum_{k=0}^{\lfloor (t-1)/d_{im} \rfloor} y_{i,m,(t-1-kd_{im})} \right) \leq \\ &\sum_{m \in M_j} \sum_{t=1}^{D} t \cdot \left(\sum_{k=0}^{\lfloor t/d_{jm} \rfloor} y_{j,m,(t-kd_{jm})} - \sum_{k=0}^{\lfloor (t-1)/d_{jm} \rfloor} y_{i,m,(t-1-kd_{jm})} \right) \qquad \forall (i,j) \in E \end{aligned} \tag{40}$$

$$\sum_{m \in M_i} r_{imk}^n \cdot e_{im} \leq a_k^n \quad \forall k \in \mathcal{R}^n \tag{41}$$

$$\sum_{i \in A} \sum_{m \in M_i} r_{imk} \cdot y_{imt} \leq a_k \quad \forall k \in \mathcal{R}, t = 0, \ldots, D \tag{42}$$

$$a_k \geq 0 \quad \forall k \in \mathcal{R} \qquad a_k^n \geq 0 \quad \forall k \in \mathcal{R}^n \tag{43}$$

$$e_{im} \in \{0, 1\} \quad \forall i \in A, \forall m \in M_i \tag{44}$$

$$z_{imt} \in \{0, 1\} \quad \forall i \in A, \forall m \in M_i, t = 1, \ldots, D. \tag{45}$$

With (36), we ensure that each activity is assigned to at least one period where it is processed in its chosen mode. The equality in (37) limits the mode choice to exactly one mode per activity. The constraints in (38) have two functions: First, they force the on/off variables corresponding to the modes that are not chosen to be 0. Second, together with (39) they ensure that activity i is executed for exactly d_{im} periods without interruption if mode m is chosen. The precedence relations are modelled in (40) in the aggregated version. The nonrenewable and renewable resource constraints are displayed in (41) and (42), respectively. Finally, the decision variables are defined in (43)–(45). Again, we can also formulate the precedence constraints in a disaggregated way (see (46)).

$$\sum_{m \in M_i} \sum_{k=0}^{\lfloor \tau / d_{im} \rfloor} y_{i,m,(\tau-(k+1)d_{im})} \geq \sum_{m \in M_j} \sum_{k=0}^{\lfloor \tau / d_{jm} \rfloor} y_{j,m,(\tau-kd_{jm})} \quad \forall (i,j) \in E, t = 1, \ldots, D. \tag{46}$$

The disaggregated version of the on/off-based formulation (OODDT) uses constraints (35)–(39),(41)–(45) and (46).

Artigues (2017) calls PDDT, SDDT and OODDT the strong formulations as their LP relaxation is superior to the LP relaxation of PDT, SDT and OODT (these three are called weak formulations). However, due to for example more memory allocation or other reasons, the weaker formulations can have a better performance when solved as mixed-integer linear program as we see in Sect. 6.

Appendix 1.3: Constraint programming formulation

The CP formulation of the MRIP is displayed in (47)–(54). To model the MRIP with CP, we use so-called *interval* variables that define the start and finish of activities. With the keyword *optional*, we specify that an interval variable is not mandatory (applied for the mode interval variables in (54)) and the keyword *size* defines the length of the interval (i.e. the difference of finish and start of an interval). We connect the interval variable *act[i]* of activity $i \in A$ with the respective mode interval variables *mode[i,m]* using the expression `alternative`. This expression ensures that exactly one of the mode intervals is present and the start and finish times of *act[i]* coincide with the chosen mode interval (see (48)). The precedence relations are modelled in (49) and the nonrenewable resource constraints in (50). For the renewable resource, we utilise a

so-called *cumulative* function $rUse_k$ together with the `pulse` operator which sums up the renewable resource requirement for the periods where the mode interval variable is in process. In (52), we define the resource allocation variables for both the renewable and nonrenewable resources. They are similar to the ones of the MIP formulation presented above and are also used in the objective function (47).

$$\min \sum_{k \in \mathcal{R}} c_k \cdot a_k + \sum_{k \in \mathcal{R}^n} c_k^n \cdot a_k^n \tag{47}$$

$$s.t. \quad \texttt{alternative}(act[i], \{mode[i,m] : m \in M_i\}) \quad \forall i \in A \tag{48}$$

$$\texttt{endBeforeStart}(act[i], act[j]) \qquad \forall (i,j) \in E \tag{49}$$

$$\sum_{i \in A} \sum_{m \in M_i} \texttt{presenceOf}(mode[i,m]) \cdot r_{imk} \leq a_k^n \qquad \forall k \in \mathcal{R}^n \tag{50}$$

$$rUse_k = \sum_{i \in A} \sum_{m \in M_i} \texttt{pulse}(mode[i,m], r_{imk}) \leq a_k \qquad \forall k \in \mathcal{R} \tag{51}$$

$$a_k \geq 0, \quad a_k^n \geq 0 \qquad \forall k \in \mathcal{R} \tag{52}$$

$$\texttt{interval}act[i] \qquad \forall i \in A \tag{53}$$

$$\texttt{interval}mode[i,m] \textit{ optional size } d_{im} \quad \forall i \in A, \forall m \in M_i. \tag{54}$$

References

Artigues C (2017) On the strength of time-indexed formulations for the resource-constrained project scheduling problem. Oper Res Lett 45(2):154–159. https://doi.org/10.1016/j.orl.2017.02.001

Bartels JH (2009) Anwendung von Methoden der ressourcenbeschränkten Projektplanung mit multiplen Ausführungsmodi in der betriebswirtschaftlichen Praxis: Rückbauplanung für Kernkraftwerke und Versuchsträgerplanung in Automobilentwicklungsprojekten. Springer, Berlin

Bartels JH, Zimmermann J (2009) Scheduling tests in automotive R&D projects. Eur J Oper Res 193(3):805–819. https://doi.org/10.1016/j.ejor.2007.11.010

Bartels JH, Zimmermann J (2015) Scheduling tests in automotive R&D projects using a genetic algorithm. In: Schwindt C, Zimmermann J (eds) Handbook on project management and scheduling, vol 2. Springer, Cham, pp 1157–1185. https://doi.org/10.1007/978-3-319-05915-0_22

Bianco L, Caramia M, Giordani S (2016) Resource levelling in project scheduling with generalized precedence relationships and variable execution intensities. OR Spectrum 38(2):405–425. https://doi.org/10.1007/s00291-016-0435-1

Boctor FF (1993) Heuristics for scheduling projects with resource restrictions and several resource-duration modes. Int J Prod Res 31(11):2547–2558. https://doi.org/10.1080/00207549308956882

Cochran WG (1952) The $\chi 2$ test of goodness of fit. Ann Math Stat 1:315–345

Colak E, Azizoglu M (2014) A resource investment problem with time/resource trade-offs. J Oper Res Soc 65(5):777–790. https://doi.org/10.1057/jors.2013.46

Coughlan ET, Lübbecke ME, Schulz J (2015) A branch-price-and-cut algorithm for multi-mode resource leveling. Eur J Oper Res 245(1):70–80. https://doi.org/10.1016/j.ejor.2015.02.043
Deckro RF, Hebert JE (1989) Resource constrained project crashing. Omega 17(1):69–79. https://doi.org/10.1016/0305-0483(89)90022-4
Demeulemeester E (1995) Minimizing resource availability costs in time-limited project networks. Manag Sci 41(10):1590–1598. https://doi.org/10.1287/mnsc.41.10.1590
Demeulemeester E, Vanhoucke M, Herroelen W (2003) RanGen: a random network generator for activity-on-the-node networks. J Schedul 6(1):17–38. https://doi.org/10.1023/A:1022283403119
Drexl A, Kimms A (2001) Optimization guided lower and upper bounds for the resource investment problem. J Oper Res Soc 52(3):340–351. https://doi.org/10.1057/palgrave.jors.2601099
Geiger MJ (2017) A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem. Eur J Oper Res 256(3):729–741. https://doi.org/10.1016/j.ejor.2016.07.024
Gerhards P, Stürck C (2018) A hybrid metaheuristic for the multi-mode resource investment problem with tardiness penalty. In: Fink A, Fügenschuh A, Geiger MJ (eds) Operations Research Proceedings 2016. Springer, Cham, pp 515–520. https://doi.org/10.1007/978-3-319-55702-1_68
Hsu CC, Kim DS (2005) A new heuristic for the multi-mode resource investment problem. J Oper Res Soc 56(4):406–413. https://doi.org/10.1057/palgrave.jors.2601827
Józefowska J, Mika M, Rózycki R, Waligóra G, Weglarz J (2001) Simulated annealing for multi-mode resource-constrained project scheduling. Ann Oper Res 102(1–4):137–155
Kelley JE (1963) The critical-path method: resources planning and scheduling. Ind Schedul 13(1):347–365
Klein R, Scholl A (1999) Computing lower bounds by destructive improvement: an application to resource-constrained project scheduling. Eur J Oper Res 112(2):322–346. https://doi.org/10.1016/S0377-2217(97)00442-6
Kolisch R (1996) Efficient priority rules for the resource-constrained project scheduling problem. J Oper Manag 14(3):179–192. https://doi.org/10.1016/0272-6963(95)00032-1
Kolisch R, Sprecher A (1997) PSPLIB—a project scheduling problem library: OR software—ORSEP operations research software exchange program. Eur J Oper Res 96(1):205–216. https://doi.org/10.1016/S0377-2217(96)00170-1
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. https://doi.org/10.1287/mnsc.41.10.1693
Kreter S, Schutt A, Stuckey PJ, Zimmermann J (2018) Mixed-integer linear programming and constraint programming formulations for solving resource availability cost problems. Eur J Oper Res 266(2):472–486. https://doi.org/10.1016/j.ejor.2017.10.014
Laborie P, Rogerie J, Shaw P, Vilím P (2018) IBM ILOG CP optimizer for scheduling. Constraints 23(2):210–250. https://doi.org/10.1007/s10601-018-9281-x
López-Ibáñez M, Dubois-Lacoste J, Cáceres LP, Birattari M, Stützle T (2016) The irace package: iterated racing for automatic algorithm configuration. Oper Res Perspect 3:43–58. https://doi.org/10.1016/j.orp.2016.09.002
Mastor AA (1970) An experimental investigation and comparative evaluation of production line balancing techniques. Manag Sci 16(11):728–746. https://doi.org/10.1287/mnsc.16.11.728
Meng H, Wang B, Nie Y, Xia X, Zhang X (2016) A scatter search hybrid algorithm for resource availability cost problem. In: Harmony search algorithm. Springer, Berlin, pp 39–51. https://doi.org/10.1007/978-3-662-47926-1_5
Möhring RH (1984) Minimizing costs of resource requirements in project networks subject to a fixed completion time. Oper Res 32(1):89–120. https://doi.org/10.1287/opre.32.1.89
Najafi AA, Azimi F (2009) A priority rule-based heuristic for resource investment project scheduling problem with discounted cash flows and tardiness penalties. Math Problems Eng 2009:1. https://doi.org/10.1155/2009/106425
Najafi AA, Niaki STA (2006) A genetic algorithm for resource investment problem with discounted cash flows. Appl Math Comput 183(2):1057–1070. https://doi.org/10.1016/j.amc.2006.05.118
Najafi AA, Niaki STA, Shahsavar M (2009) A parameter-tuned genetic algorithm for the resource investment problem with discounted cash flows and generalized precedence relations. Comput Oper Res 36(11):2994–3001. https://doi.org/10.1155/2009/106425

Neumann K, Schwindt C, Zimmermann J (2003) Project scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions. Springer, Berlin

Qi JJ, Liu YJ, Jiang P, Guo B (2015) Schedule generation scheme for solving multi-mode resource availability cost problem by modified particle swarm optimization. J Schedul 18(3):285–298. https://doi.org/10.1007/s10951-014-0374-0

Ranjbar M, Kianfar F, Shadrokh S (2008) Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. Appl Math Comput 196(2):879–888. https://doi.org/10.1016/j.amc.2007.07.022

Rieck J, Zimmermann J (2015) Exact methods for resource leveling problems. In: Schwindt C, Zimmermann J (eds) Handbook on project management and scheduling, vol 1. Springer, Cham, pp 361–387. https://doi.org/10.1007/978-3-319-05443-8_17

Rodrigues SB, Yamashita DS (2010) An exact algorithm for minimizing resource availability costs in project scheduling. Eu J Oper Res 206(3):562–568. https://doi.org/10.1016/j.advengsoft.2010.03.002

Rodrigues SB, Yamashita DS (2015) Exact methods for the resource availability cost problem. In: Schwindt C, Zimmermann J (eds) Handbook on project management and scheduling, vol 1. Springer, Cham, pp 319–338. https://doi.org/10.1007/978-3-319-05443-8_15

Schnell A, Hartl RF (2016) On the efficient modeling and solution of the multi-mode resource-constrained project scheduling problem with generalized precedence relations. OR Spectrum 38(2):283–303. https://doi.org/10.1007/s00291-015-0419-6

Schwindt C (1998) Generation of resource constrained project scheduling problems subject to temporal constraints. Inst. für Wirtschaftstheorie und Operations-Research

Shadrokh S, Kianfar F (2007) A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. Eur J Oper Res 181(1):86–101. https://doi.org/10.1016/j.ejor.2006.03.056

Sprecher A, Drexl A (1998) Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. Eur J Oper Res 107(2):431–450. https://doi.org/10.1016/S0377-2217(97)00348-2

Talbot FB (1982) Resource-constrained project scheduling with time-resource tradeoffs: the nonpreemptive case. Manag Sci 28(10):1197–1210. https://doi.org/10.1287/mnsc.28.10.1197

Van Peteghem V, Vanhoucke M (2013) An artificial immune system algorithm for the resource availability cost problem. Flex Serv Manuf J 25(1–2):122–144. https://doi.org/10.1007/s10696-011-9117-0

Van Peteghem V, Vanhoucke M (2014) An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. Eur J Oper Res 235(1):62–72. https://doi.org/10.1016/j.ejor.2013.10.012

Van Peteghem V, Vanhoucke M (2015) Heuristic methods for the resource availability cost problem. In: Schwindt C, Zimmermann J (eds) Handbook on project management and scheduling, vol 1. Springer, Cham, pp 339–359. https://doi.org/10.1007/978-3-319-05443-8_16

Verbeeck C, Van Peteghem V, Vanhoucke M, Vansteenwegen P, Aghezzaf EH (2017) A metaheuristic solution approach for the time-constrained project scheduling problem. OR Spectrum 39(2):353–371. https://doi.org/10.1007/s00291-016-0458-7

Yamashita DS, Morabito R (2009) A note on time/cost tradeoff curve generation for project scheduling with multi-mode resource availability costs. Int J Oper Res 5(4):429–444. https://doi.org/10.1504/IJOR.2009.025702

Yamashita DS, Armentano VA, Laguna M (2006) Scatter search for project scheduling with resource availability cost. Eur J Oper Res 169(2):623–637. https://doi.org/10.1016/j.amc.2006.05.118

Yamashita DS, Armentano VA, Laguna M (2007) Robust optimization models for project scheduling with resource availability cost. J Sched 10(1):67–76. https://doi.org/10.1007/s10951-006-0326-4

Yuan X, Liu J, Hao X (2017) A moving block sequence-based evolutionary algorithm for resource investment project scheduling problems. Big Data Inf Anal 2(1):39–58. https://doi.org/10.3934/bdia.2017007

Zhu X, Ruiz R, Li S, Li X (2017) An effective heuristic for project scheduling with resource availability cost. Eur J Oper Res 257(3):746–762. https://doi.org/10.1016/j.ejor.2016.08.049