European Journal of Operational Research 303 (2022) 1071–1087

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

On the integration of diverging material flows into resource-constrained project scheduling☆

Marco Gehring*, Rebekka Volk, Frank Schultmann

Karlsruhe Institute of Technology, Institute for Industrial Production (IIP), Hertzstr. 16, Karlsruhe 76187, Germany

ARTICLE INFO

Article history:
Received 12 April 2021
Accepted 24 March 2022
Available online 29 March 2022

Keywords:
Project scheduling
Material flows
Storage facilities
Cumulative resources

ABSTRACT

This study deals with an extension of the resource-constrained project scheduling problem (RCPSP) by constraints on material flows released during the execution of project activities. These constraints arise from limited processing capacities for materials and maximum inventories of intermediate storage facilities. Production scheduling problems with converging material flows have been studied extensively. However, this is the first project scheduling problem integrating diverging material flows typically observed in dismantling projects, e.g., building deconstruction, power plant decommissioning, or battery/car decommissioning. Diverging material flows do not directly impact the project planning but only impose delays in the case of congestion. We model material flows by using operations that represent the processing of materials, and cumulative resources that represent storage facilities. As a method for efficiently generating starting solutions, we propose a schedule generation scheme tailored to the particular precedence structure of such problems. Furthermore, we extensively study the schedule generation scheme's performance on generated test instances and compare it to the constraint programming solver IBM ILOG CP Optimizer. It turns out that the solution quality strongly depends on the employed model and that neither of the two solution methods is generally superior.

1. Introduction

Diverging material flows can impose delays on a project schedule in the case of congestion. Existing scheduling models are unable to reflect these delays during the planning phase of a project. In this study, we introduce a new scheduling model that integrates diverging material flows into resource-constrained project scheduling.

We consider the problem of scheduling a project with the objective of minimizing the project makespan. The project is split into several non-interruptible activities. Precedence constraints specify certain activities that cannot start before other activities have been completed. The critical path method (Kelley, 1961) is a well-known technique to solve these problems in polynomial time (cf. Neumann, Schwindt, & Zimmermann, 2003) under the assumption of sufficiently available resources. This assumption is usually not appropriate for many real-life applications. Therefore, extensive research has been devoted to the *resource-constrained project scheduling problem (RCPSP)*. This problem involves availability constraints on renewable resources, which are required for the execution of activities (cf. e.g., Brucker, Drexl, Möhring, Neumann, & Pesch, 1999 for a comprehensive review). These availability constraints can impose a delay on the project makespan. Finding an optimal schedule that is feasible both regarding precedence constraints and availability constraints on renewable resources has been proven to be strongly NP-hard (Blazewicz, Lenstra, & Kan, 1983).

Analogously, we consider bottlenecks when processing material flows released by activities as another type of constraint imposing a delay on the project makespan. We refer to them as *material flow constraints*. This problem is motivated by ongoing research projects about scheduling the dismantling of nuclear power plants and wind turbines. Such projects are characterized by *diverging material flows* that are released when executing an activity and can have a back coupling effect on the original project schedule. The counterpart, *converging material flows*, which are typically observed in production planning, are not under study since these settings have been addressed extensively in the literature (cf. Section 3). We refer to Neumann & Schwindt (1997) for a definition of a con-

☆ This work was supported by the Federal Ministry of Education and Research of Germany (BMBF) in the NukPlaRStoR project (Grant number: 15S9414A). The sole responsibility of this publication lies with the authors. The BMBF is not responsible for any results or recommendations stated by authors. We thank the anonymous reviewers for their constructive comments that greatly improved the paper.

* Corresponding author.
E-mail address: marco.gehring@kit.edu (M. Gehring).

https://doi.org/10.1016/j.ejor.2022.03.047

vergent product structure which gives rise to converging material flows. The processing of diverging material flows is not considered in the objective function but can only indirectly curtail the solution space. For example, in nuclear dismantling projects, the material processing is typically outsourced to so-called "material processing centers" which are responsible for, e.g., cutting, cleaning, handling, and packaging the wastes (EnBW Kernkraft GmbH, 2018). Due to restricted data access in this sensitive industry, we do not further consider the problem from a nuclear dismantling perspective but from a more theoretical perspective.

After having been released by an activity, a diverging material flow traverses a network of *processing steps* (e.g., milling, segmenting, transferring, decontaminating) and intermediate *storage facilities* until it reaches the sink (e.g., recycling facility, landfill). As long as less material is released than can be processed, activities are scheduled at the earliest times where precedence constraints and availability constraints on renewable resources are satisfied. However, as soon as an overload of *processing capacity* occurs, storage facilities must start buffering the material. If this continues until the *maximum inventory* of storage facilities is exceeded, the schedule gets affected and activities must be delayed, which can eventually result in prolonging the project's makespan. Hence, the entirety of processing steps and storage facilities with their limitations form the material flow constraints.

To make our problem setting compatible with existing concepts from scheduling literature, we systematically integrate material flows into the well-known structure of the RCPSP. Therefore, we introduce *operations* as separately scheduled entities and harness the *cumulative resource type* (cf. e.g., Schwindt, 1999; Neumann & Schwindt, 2002) for modeling storage facilities with maximum inventories. In contrast to renewable resources, the availability of cumulative resources depends on all previous requirements. Thus, they are particularly suitable for modeling storages. Compared to solution methods for related problems, such as branch-and-bound techniques (cf. Neumann & Schwindt, 2002), our approach exploits special characteristics of diverging material flows.

Thus, our main contributions are:

- We introduce a novel project scheduling problem taking into account diverging material flows that have a back coupling effect on the project planning. First, we conceptually formulate the problem, and second, we formulate the problem using the structure of the RCPSP.
- We present a schedule generation scheme (SGS) that serves the following purposes: It helps to understand the problem's structural properties. It provides insight into the feasibility of a problem instance. It can be integrated into metaheuristics and used to obtain good starting solutions. And, it facilitates the performance assessment of other solution methods by providing a benchmark.
- We extensively study the tractability of the problem using generated test instances with different characteristics. Therefore, we compare several modeling variants and apply the SGS and the constraint programming solver IBM ILOG CP Optimizer as solution methods. Regarding the latter, we provide the first known performance assessment for the RCPSP with cumulative resources.

The paper's remainder is organized as follows: In Section 2, we introduce the developed material flow model and illustrate the setting using an exemplary material flow. Section 3 presents a literature review to relate to existing studies and to outline the main research gaps. Our novel project scheduling problem is termed *resource-and-material-flow-constrained project scheduling problem (RMCPSP)* and stated formally in Section 4. Section 5 outlines a transformation scheme that allows for remodeling the RMCPSP as a RCPSP with cumulative resources. The SGS is presented in Section 6, combining existing concepts from scheduling literature and taking into account the problem's structural properties. Section 7 deals with modeling variants. Section 8 is devoted to computational experiments and their discussion. Section 9 provides a summary and conclusions.

2. Material flow model

Since the RCPSP is a classical discrete optimization problem, we also assume the diverging material flows to be discrete. Formally, we investigate *material units* (e.g., containers or tons). Material units are considered as homogeneous in terms of storage volume. Typically, a material unit should be chosen as the greatest common divisor of all material flow values observed during the project.

We use two types of parameters to describe material flows: First, a set of *material flow paths* and second, the *number of material units released by each activity, traversing each material flow path* (see Section 4 for a formal definition). Material flow paths are sequences of processing steps and storage facilities, each terminating with the sink. Since the inventory of sinks is unconstrained, it is sufficient to use a single sink for all material flows. The same processing steps and storage facilities may repeatedly occur in a material flow path. Without loss of generality, in each material flow path processing steps and storage facilities are arranged in alternation. And, each material flow path starts with a storage facility, in which the activity releases the material units. If there exists a processing step with a zero-wait condition, we set the capacity of the preceding storage facility to zero.

Consider the following illustrative example. A material flow released by activity 1 consists of five material units. The material flow is described as follows, where P*x* are processing steps, S*x* are storage facilities, and s is the sink:

- Material flow path 1 = $(\mathtt{S1}, \mathtt{P1}, \mathtt{S1}, \mathtt{P2}, \mathtt{S3}, \mathtt{P3}, \mathtt{S5}, \mathtt{P7}, s)$; traversed by one material unit.
- Material flow path 2 = $(\mathtt{S4}, \mathtt{P4}, \mathtt{S5}, \mathtt{P5}, \mathtt{S5}, \mathtt{P8}, s)$; traversed by two material units.
- Material flow path 3 = $(\mathtt{S4}, \mathtt{P4}, \mathtt{S5}, \mathtt{P8}, s)$; traversed by one material unit.
- Material flow path 4 = $(\mathtt{S2}, \mathtt{P6}, s)$; traversed by one material unit.

Fig. 1 depicts the material flow described by these paths as a flow network. When being released by activity 1, one material unit is replenished into storage facility S1, three material units into S4, and one unit into S2. Then, the material units are eligible for the first processing steps P1, P4, or P6 as described in the material flow paths. This means they are depleted from the upstream storage facilities S1, S4, or S2 at the start of processing and processed for a given duration. After processing, one unit reaches the sink and the other four units are replenished into the downstream storage facilities S1 and S5, respectively. The processing continues and thus, all material units are incrementally moved towards the sink while creating new space for material units from other activities.

The flow network in Fig. 1 has divergent, convergent, and cyclical parts. In general, materials can flow arbitrarily between different storage facilities. When we schedule the processing of the described material flow, however, we observe a diverging structure as the materials are no longer merged and processed as one unit throughout the planning horizon. This significantly reduces the complexity of the problem, as we will show in the following elaboration. This model is particularly suitable for practical applications with material flows that have a back coupling effect on the planning due to processing and storage bottlenecks, e.g., dismantling projects.

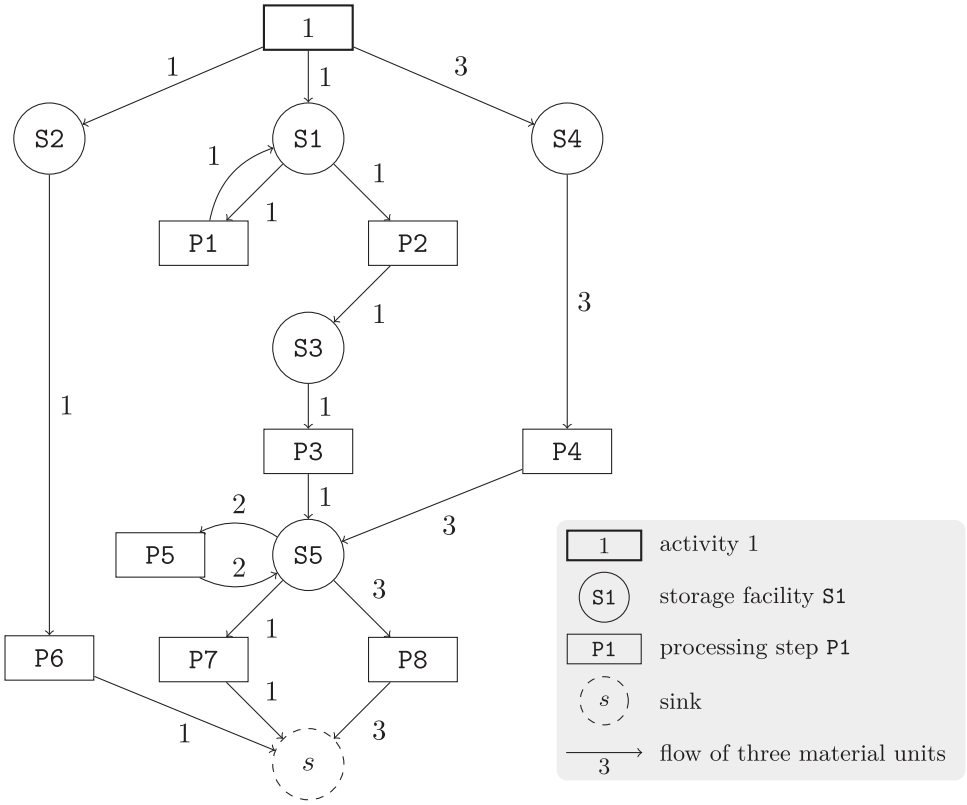

Fig. 1. Exemplary flow network.

Without loss of generality, we decide for the following modeling concepts:

(1) The limited processing capacity is modeled through renewable resources. More precisely, we define that applying a processing step to a material unit requires specific amounts of renewable resources. The pool of renewable resources is the same for both activities and processing steps. Thus, project activities and the material flows may compete against the same renewable resources such as workers or tools.
(2) The processing of material units released by an activity is interruptible down to the level of a single material unit and a discrete time period. We can treat the non-interruptible case with the same concepts and refer to it as the aggregated modeling variant in Section 7.
(3) All material units, which are released by a certain activity i and which feature the same material flow path emerge uniformly distributed over the execution time of i (cf. e.g., Neumann, Schwindt, & Trautmann, 2005; Sourd & Rogerie, 2005). This is derived from the assumption that the work progress of activities is linear in time and thus material emerges at equal time intervals (see Section 4 for a formal definition). If necessary, our proposed SGS can also solve problems with other temporal distributions of the emerging times of material units.

Furthermore, we derive two assumptions from the practical problem setting:

(1) The minimum inventory of each storage facility is zero. This is inherent to the studied setting since all material flows are intended to end up in a sink.
(2) There are no maximum storage times (also called “shelf-life times” by Schwindt & Trautmann (2000)) since we do not deal with perishable products.

3. Literature review

Most papers in scheduling literature incorporating material flows address problems arising in production planning, e.g., Blömer & Günther (1998), Blömer & Günther (2000), Kallrath (2002), Burkard & Hatzl (2005), Burkard & Hatzl (2006), Belaid, T'kindt, & Esswein (2012), Boysen, Bock, & Fliedner (2013), Baumann & Trautmann (2014), Briskorn & Zeise (2019). A network representation of chemical processes called *State-Task-Network (STN)*, which has several similarities to our representation in Fig. 1, is proposed by Kondili, Pantelides, & Sargent (1993). Storage facilities are regarded as states, whereas processing steps correspond to tasks. However, the STN representation requires that all material flows entering the same state (i.e., the same storage facility) are of the same quality since states usually refer to tanks in practice. This is not a suitable assumption for our problem setting with diverging material flows, where we keep track of every single material unit.

In the area of project scheduling, there is only a limited number of papers taking into account material flows. Schwindt & Trautmann (2000) present an integration of inventory constraints into the framework of the RCPSP for production settings, where material flows are processed as predetermined and discrete quantities called *batches*. Schwindt, Fink, & Trautmann (2007) tackle the same problem using a priority rule method for generating feasible schedules. In our interruptible setting, we would identify the processing of each single material unit as a batch. Neumann et al. (2005) propose an extended model including the case of *continuous material flows* processed at finite rates over the duration of an activity. Voß & Witt (2007) model a steel manufacturing problem utilizing a multi-mode RCPSP and apply priority rules for solving real-world instances. They do not take storage restrictions into account. Besides, their definition of the term “batch” differs from the definition by Schwindt & Trautmann (2000) since they regard

a batch as a series of processing steps that is executed on a machine without changing its setup.

In all application-oriented publications cited so far, material flows are an integral part of the project, and delaying critical processing steps would directly prolong the project makespan. For example, Schwindt & Trautmann (2000) define a demand vector indicating the requirements for final products to be produced with different input materials to minimize the total production makespan. Boysen et al. (2013) are concerned with the problem of scheduling a single machine so that external demand events occurring at fixed times over the planning horizon are satisfied. In our problem setting, the processing of material flows is not an integral part of the project, but a secondary planning level. More precisely, material processing cannot lie on the critical path, but only indirectly impacts the project's makespan in the case of storage overload. To the best of our knowledge, this case has not been addressed in the literature so far.

In the following, we provide a brief overview of solution methods for scheduling problems with storage constraints. Schwindt (1999) and Neumann & Schwindt (2002) introduced the cumulative resource type as a generalization of renewable and nonrenewable resources. Both Schwindt & Trautmann (2000) and Neumann et al. (2005) model storage facilities in processing industries as cumulative resources. To generate feasible solutions for the RCPSP with cumulative resources, they relax minimum and maximum inventory constraints and iteratively add temporal constraints between activities to resolve remaining inventory shortages or excesses. By integrating this generation scheme into an exact (truncated) branch-and-bound procedure, they compute optimal (heuristic) solutions for different test instances. It is well-known that branch-and-bound techniques perform poorly for large instances (cf. Franck, Neumann, & Schwindt, 2001). Neumann et al. (2005) show that their algorithm can handle instances comprising 750 operations with batch material flows and up to 90 operations with continuous material flows. Laborie (2003) presents an alternative approach to solving these problems using constraint programming techniques. It is extended to the case of continuous material flows by Sourd & Rogerie (2005). Carlier, Moukrim, & Xu (2009) examine a list scheduling algorithm for tackling a particular variant of the problem considered by Neumann & Schwindt (2002) and Laborie (2003) with a single storage facility. Carlier, Moukrim, & Sahli (2018) provide lower bounds to the latter problem. Koné, Artigues, Lopez, & Mongeau (2013) integrate cumulative resources into mixed-integer linear programming (MILP) models. MILP formulations are flexible for extensions, while the solution method remains unchanged. However, the performance is weak for large instances compared to specialized algorithms.

Briskorn, Choi, Lee, Leung, & Pinedo (2010) introduce a single machine scheduling problem subject to inventory constraints. Jobs replenish or deplete a central storage facility with unlimited capacity. As long as sufficient items have not been replenished into the storage, no depleting job is eligible for its execution. The problem is motivated by the scheduling of trucks that deliver and collect items at a transshipment terminal. Solution procedures and bounds for this problem, but with varying objective functions, are studied in Briskorn, Jaehn, & Pesch (2013), Briskorn & Leung (2013) and Morsy & Pesch (2015). Briskorn & Pesch (2013) consider the case of limited storage capacity and propose a heuristic solution procedure based on the variable neighborhood algorithm. Davari, Ranjbar, de Causmaecker, & Leus (2020) include release date constraints into the problem formulation by Briskorn et al. (2010).

4. Problem statement

The RMCPSP proposed in this paper deals with scheduling a project that has been broken down into a set of non-interruptible activities $i = 0, \dots, I+1$ to minimize the project makespan z. A vector of start times $S := (S_i)_{i=0,\dots,I+1}$ is called a *schedule*. Fictitious activities 0 and $I+1$ represent the start and the end of the project, i.e., $S_0 := 0$ and $z := S_{I+1}$. Parameter $d_i \in \mathbb{Z}_{\geq 0}$ indicates the duration (in periods) of each activity i, where $d_0 = d_{I+1} := 0$ holds.

A set of *precedence relations* $E \subset \{0, \dots, I+1\}^2$ is defined on pairs of activities, where each precedence relation $(i, i') \in E$ implies a temporal constraint of the finish-to-start type termed *precedence constraint*; i.e., activity i' must not start before the completion of activity i. Let $\mathcal{I}_{\text{RMCPSP}}$ denote an instance of the RMCPSP. A schedule S to $\mathcal{I}_{\text{RMCPSP}}$ is *time-feasible* if and only if it satisfies precedence constraints

$$S_{i'} \geq S_i + d_i \quad (\forall (i, i') \in E). \tag{1}$$

We assume that precedence relations are appropriately contained in E in order to make sure that fictitious activities 0 and $I+1$ uniquely represent the start and the end of the project.

The set of renewable resources available in our project is denoted by $\mathcal{R}^\alpha$. The maximum availability of each renewable resource $k \in \mathcal{R}^\alpha$ is given by $R_k^\alpha \in \mathbb{Z}_{\geq 0}$. An activity i requires $r_{ik}^\alpha \in \mathbb{Z}_{\geq 0}$ units of renewable resource $k \in \mathcal{R}^\alpha$ during its execution. After completion, these units can be used by other activities. Given a schedule S, the so-called *active set*

$$\mathcal{A}^\alpha(S, t) := \{i \in \{0, \dots, I+1\} \mid S_i \leq t < S_i + d_i\} \tag{2}$$

comprises all activities executed at a time $t \geq 0$ (cf. Neumann, Nübel, & Schwindt, 2000). Then, the overall amount of renewable resource k required by project activities at a time $t \geq 0$ is

$$r_k^\alpha(S, t) := \sum_{i \in \mathcal{A}^\alpha(S,t)} r_{ik}^\alpha. \tag{3}$$

With regard to material flows, the set of storage facilities is denoted by $\mathcal{R}^\gamma$, the set of processing steps by $\mathcal{P}$, and the sink by s. The maximum inventory of each storage facility $k \in \mathcal{R}^\gamma$ is prescribed by parameter $R_k^\gamma \in \mathbb{Z}_{\geq 0}$. Analogous to project activities, each processing step $p \in \mathcal{P}$ requires $r_{pk}^\alpha \in \mathbb{Z}_{\geq 0}$ units of renewable resource $k \in \mathcal{R}^\alpha$ during every execution period. The overall amount of a renewable resource $k \in \mathcal{R}^\alpha$ required by all processing steps at a time $t \geq 0$ is denoted by function $\Phi_k^\alpha(t)$. A schedule S to $\mathcal{I}_{\text{RMCPSP}}$ is *renewable-resource-feasible* if and only if it satisfies *availability constraints on renewable resources*

$$r_k^\alpha(S, t) + \Phi_k^\alpha(t) \leq R_k^\alpha \quad (\forall k \in \mathcal{R}^\alpha, t \geq 0). \tag{4}$$

Note that function $\Phi_k^\alpha(t)$ represents the results from the secondary planning level for material flow processing. We do not provide a mechanism for evaluating it in this conceptual problem statement, but refer to Section 5 where material flows are integrated into the project planning.

The set of all pairwise distinct material flow paths observed during the project is denoted by W. The first storage facility in a path $w \in W$ is referred to as $k_1(w)$, the first processing step as $p_1(w)$, and so on. Now, let us assume that each activity i releases $f_{iw} \in \mathbb{Z}_{\geq 0}$ material units traversing the material flow path $w \in W$, where $f_{0w} = f_{I+1w} := 0$ holds for each $w \in W$. As mentioned in Section 2, all f_{iw} material units emerge uniformly distributed over the execution time of i. Following the approach by Neumann et al. (2005) for continuous material flows, we therefore introduce function

$$x_i(S, t) := \begin{cases} 0 & \text{if } t < S_i, \\ 1 & \text{if } t \geq S_i + d_i, \\ (t - S_i)/d_i & \text{otherwise,} \end{cases} \tag{5}$$

for denoting the portion of activity i that has been completed at a time $t \geq 0$, given a schedule S. The active set

$$\mathcal{A}^\gamma(S, t) := \{i \in \{0, \dots, I+1\} \mid 0 \leq S_i \leq t\} \tag{6}$$

Table 1
Parameters introduced for $\mathcal{I}_{\text{RMCPSP}}$.

Notation	Denotation
$i = 0, \dots, I+1$	(project) activities; I denotes the number of non-fictitious activities
$d_i \in \mathbb{Z}_{\geq 0}$	duration of activity i
$(i, i') \in E$	precedence relation
$k \in \mathcal{R}^\alpha$	renewable resource
$r^\alpha_{ik} \in \mathbb{Z}_{\geq 0}$	number of units of renewable resource $k \in \mathcal{R}^\alpha$ required by activity i
$R^\alpha_k \in \mathbb{Z}_{\geq 0}$	maximum availability of renewable resource $k \in \mathcal{R}^\alpha$
$p \in \mathcal{P}$	processing step
$r^\alpha_{pk} \in \mathbb{Z}_{\geq 0}$	number of units of renewable resource $k \in \mathcal{R}^\alpha$ required by processing step $p \in \mathcal{P}$
$k \in \mathcal{R}^\gamma$	storage facility
$R^\gamma_k \in \mathbb{Z}_{\geq 0}$	maximum inventory of storage facility $k \in \mathcal{R}^\gamma$
$w \in W$	material flow path of a material unit released during project execution
$f_{iw} \in \mathbb{Z}_{\geq 0}$	number of material units, whose flow is described by material flow path $w \in W$, released by activity i

comprises all activities that have released material units from the project start until a time $t \geq 0$, given a schedule S. Then, the overall amount of material units released into a storage facility $k \in \mathcal{R}^\gamma$ at a time $t \geq 0$ is

$$f^\gamma_k(S,t) := \sum_{i \in \mathcal{A}^\gamma(S,t)} \sum_{w \in W | k_1(w) = k} \lfloor f_{iw} x_i(S,t) \rfloor. \tag{7}$$

In the second sum, we only consider material flow paths $w \in W$ that start with storage facility k. The term $\lfloor f_{iw} x_i(S,t) \rfloor$ corresponds to the number of material units described by material flow path w and released by activity i at time t. The floor function ensures that a material unit is only considered as released when it is completely available at time t. Here, we differ from continuous material flows. The value of $f^\gamma_k(S,t)$ is constrained by the availability of storage facility $k \in \mathcal{R}^\gamma$ at time t. This is denoted by function $\Gamma^\gamma_k(t)$, which depends first on k's maximum inventory R^γ_k and second on how fast material units previously released are processed and moved towards the sink (a mechanism for evaluating $\Gamma^\gamma_k(t)$ is provided in Section 5). A schedule S to $\mathcal{I}_{\text{RMCPSP}}$ is *material-flow-feasible* if and only if it satisfies *material flow constraints*

$$f^\gamma_k(S,t) \leq \Gamma^\gamma_k(t) \quad (\forall k \in \mathcal{R}^\gamma, t \geq 0). \tag{8}$$

Summing up, the RMCPSP is conceptually formulated as follows:

$$\min_S \quad z := S_{I+1} \tag{9a}$$

$$\text{subject to} \quad S_{i'} \geq S_i + d_i \quad (\forall (i,i') \in E); \tag{9b}$$

$$r^\alpha_k(S,t) + \Phi^\alpha_k(t) \leq R^\alpha_k \quad (\forall k \in \mathcal{R}^\alpha, t \geq 0); \tag{9c}$$

$$f^\gamma_k(S,t) \leq \Gamma^\gamma_k(t) \quad (\forall k \in \mathcal{R}^\gamma, t \geq 0); \tag{9d}$$

$$S_0 = 0; \tag{9e}$$

$$S_i \geq 0 \quad (i = 1, \dots, I+1). \tag{9f}$$

A schedule S to $\mathcal{I}_{\text{RMCPSP}}$ is *feasible* if and only if it satisfies (9b)–(9f). Table 1 summarizes all parameters introduced for $\mathcal{I}_{\text{RMCPSP}}$.

Although R^α_k and R^γ_k are constant over time, non-constant maximum availabilities and inventories can be modeled by introducing dummy activities fixed to appropriate start times as proposed by Bartusch, Möhring, & Radermacher (1988).

5. Transformation scheme

In this section, we remodel the RMCPSP using concepts from scheduling literature, in particular, from Schwindt & Trautmann (2000) and Neumann et al. (2005). The remodeled problem is called *resource-constrained project scheduling problem with cumulative resources (RCPSP/c)*. The procedure for turning an instance $\mathcal{I}_{\text{RMCPSP}}$ into an instance $\mathcal{I}_{\text{RCPSP/c}}$ is referred to as *transformation scheme*. Algorithm 1 formally summarizes the transformation scheme. The combination of both the transformation scheme and the SGS (presented in Section 6) can be considered as one possible approach to solve the RMCPSP.

Algorithm 1: `Transformation`.

```
   Data: Instance I_RMCPSP
 1 create a new fictitious operation o_{J+1};
 2 create a new cumulative resource k ∈ R^γ for each storage
   facility k ∈ R^γ;
 3 for activity i := 0 to I + 1 do
 4     foreach material flow path w ∈ W do
 5         for material unit u := 1 to f_iw do
 6             derive a new operation j from processing step
               p_1(w);
 7             create a new release relation (i, j) ∈ E^rel and set
               d^min_ij := ⌈u · d_i / f_iw⌉;
 8             set r^α_jk := r^α_{p_1(w)k} for each k ∈ R^α;
 9             set r^γ_{jk_1(w)} := −1;
10             initialize step := 2;
11             while sink s not reached do
12                 derive a new operation j' from processing step
                   p_step(w);
13                 create a new flow-induced precedence relation
                   (j, j') ∈ E^flow;
14                 set r^α_{j'k} := r^α_{p_step(w)k} for each k ∈ R^α;
15                 set r^γ_{j'k_step(w)} := −1 and r^γ_{jk_step(w)} := r^γ_{jk_step(w)} + 1;
16                 set j := j';
17                 set step := step + 1;
18             end
19             create a new flow-induced precedence relation
               (j, o_{J+1}) ∈ E^flow;
20         end
21     end
22 end
   Result: Instance I_RCPSP/c
```

The basic idea of remodeling the RMCPSP within the framework of the RCPSP/c is to integrate the material flow planning into the project planning. To this end, the application of processing steps to material units is modeled with separately scheduled entities, termed *operations*. More precisely, for each activity i, we derive one operation j for each processing step $p_1(w), p_2(w), \dots$ in the material flow path w traversed by each material unit released by i (cf. Lines 6 and 12 of Algorithm 1). In contrast to processing steps that describe technical processing routes, each operation is a concrete

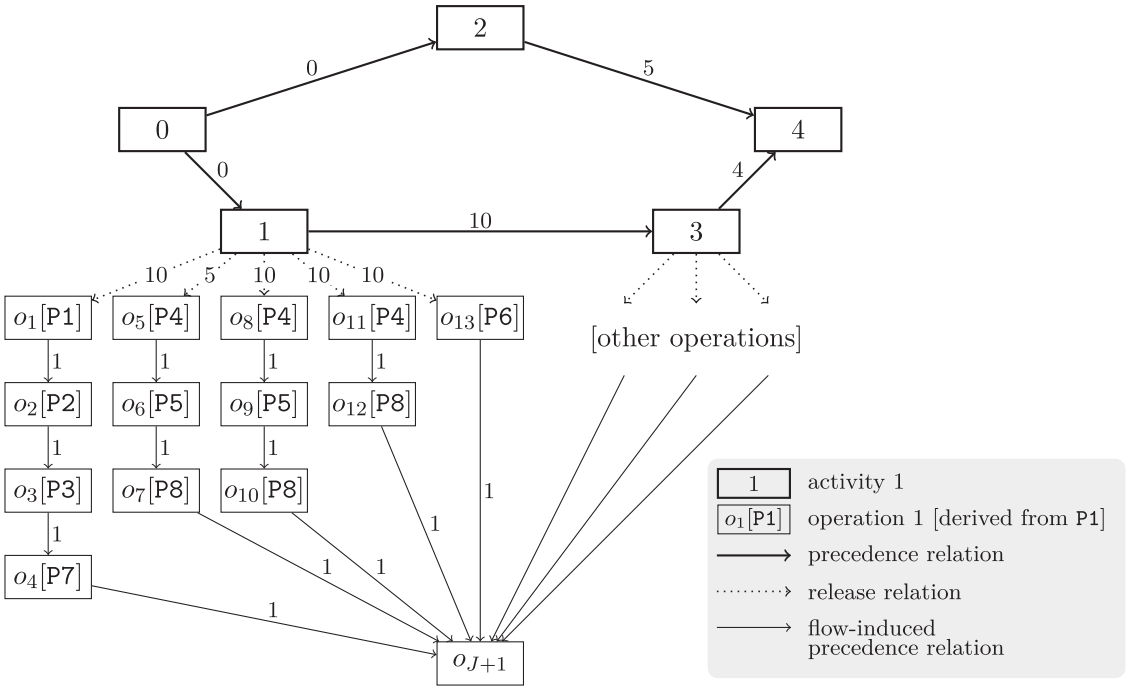

Fig. 2. Exemplary precedence network.

instance of a processing step applied to one material unit. Parameter $d_j \in \mathbb{Z}_{\geq 0}$ indicates the duration (in periods) of an operation j and must be specified a priori.

Given all operations $j = o_1, \ldots, o_{J+1}$, a schedule to the RCPSP/c is a couple (S, S^o) of a vector of start times for project activities $S := (S_i)_{i=0,\ldots,I+1}$ and a vector of start times for operations $S^o := (S^o_j)_{j=o_1,\ldots,o_{J+1}}$. Fictitious operation o_{J+1} with $d_{o_{J+1}} = 0$ represents the end of material processing. Since each operation can flexibly start after its material unit is available, the processing of material is interruptible as stated in Section 2.

Many scheduling approaches dealing with problems arising in process industries, such as Schwindt & Trautmann (2000) or Blömer & Günther (1998), do not link operations explicitly to each other. Instead, they maintain the precedences implied by the production structure just by satisfying minimum inventory constraints. However, we need an explicit link in our setting in order to ensure that processing of material units follows the prescribed and highly individual order (cf. e.g., Neumann & Schwindt, 1997). Here, such a link is modeled by a set of *flow-induced precedence relations* $E^{flow} \subset \{o_1, \ldots, o_{J+1}\}^2$, where each flow-induced precedence relation $(j, j') \in E^{flow}$ implies a temporal constraint of the finish-to-start type termed *flow-induced precedence constraint*, i.e., operation j' must not start before the completion of operation j. For each pair of immediately successive operations $j, j' \in \{o_1, \ldots, o_{J+1}\}$ on each material unit, we introduce one single flow-induced precedence relation (j, j') (cf. Line 13 of Algorithm 1). Thus, the set of flow-induced precedence relations between all operations on a material unit forms a chain. We additionally introduce flow-induced precedence relations so that fictitious operation o_{J+1} uniquely represents the end of all operations (cf. Line 19 of Algorithm 1).

Moreover, we must establish a link between the project activities and the first operation executed on each material unit after having been released by the respective activity. Such a link is modeled with a set of *release relations* $E^{rel} \subset \{0, \ldots, I+1\} \times \{o_1, \ldots, o_{J+1}\}$. Since material units can be released while project activities are still in progress (cf. Formula (7)), each release relation $(i, j) \in E^{rel}$ implies a temporal constraint of the start-to-start type with a minimum time lag $d^{min}_{ij} \in \mathbb{Z}_{\geq 0}$, i.e., operation j must not start before time $S_i + d^{min}_{ij}$. For each pair of a releasing activity $i \in \{0, \ldots, I+1\}$ and the first operation $j \in \{o_1, \ldots, o_{J+1}\}$ on each material unit, we introduce one single release relation (i, j). Our algorithm selects minimum time lags in a way that reflects the uniform emergence of material units over the activity's duration (cf. Line 7 of Algorithm 1).

A schedule (S, S^o) to $\mathcal{I}_{\text{RCPSP/c}}$ is *time-feasible* if and only if it satisfies precedence constraints (equal to (9b) in the RMCPSP)

$$S_{i'} \geq S_i + d_i \quad (\forall (i, i') \in E), \tag{10}$$

flow-induced precedence constraints

$$S^o_{j'} \geq S^o_j + d_j \quad (\forall (j, j') \in E^{flow}) \tag{11}$$

and release constraints

$$S^o_j \geq S_i + d^{min}_{ij} \quad (\forall (i, j) \in E^{rel}). \tag{12}$$

Let us consider the problem instance as a network with node set $\{0, \ldots, I+1\} \cup \{o_1, \ldots, o_{J+1}\}$ and arc set $E \cup E^{flow} \cup E^{rel}$. Arcs $(i, i') \in E$ are weighted with durations d_i, arcs $(j, j') \in E^{flow}$ with durations d_j and arcs $(i, j) \in E^{rel}$ with minimum time lags d^{min}_{ij}. As mentioned above, fictitious activity $I+1$ represents the end of the project and thus accounts for the objective value of the scheduling problem. However, the node identified with $I+1$ is not the terminal node of the network since we treat diverging material flows on a secondary planning level: None of the project activities $i = 0, \ldots, I+1$ is allowed as a (direct or transitive) successor of an operation $j = o_1, \ldots, o_{J+1}$. Particularly, fictitious activity $I+1$ is not a successor of operations. Thus, operations are not directly critical with respect to the objective function of the scheduling problem, since they cannot lie on the longest path from activity 0 to $I+1$. They only impact the objective value as soon as storage overloads are fed back to the activities' level. For illustration, Fig. 2 depicts an exemplary precedence network with $I = 3$ non-fictitious activities. Activities 1 and 3 release material units during their execution. All operations $j = o_1, \ldots, o_{13}$ are assumed to take $d_j = 1$ period. Activity 4 represents the end of the project and operation o_{J+1} represents the end of material processing. Activity 1 is identical to the activity exemplarily mentioned in Section 2 and Fig. 1. In contrast

to the flow network in Fig. 1, the precedence network in Fig. 2 reveals the diverging structure of the material flow.

With regard to renewable resources, let $O_p \subseteq \{o_1, \dots, o_J\}$ denote the set of operations derived from a processing step $p \in \mathcal{P}$. Then, each operation $j \in O_p$ requires $r^{\alpha}_{jk} := r^{\alpha}_{pk}$ units of renewable resource $k \in \mathcal{R}^{\alpha}$, i.e., the resource requirements of processing steps are handed down to the derived operations (cf. Lines 8 and 14 of Algorithm 1). Given S^o, the active set

$$\mathcal{A}^{\alpha}(S^o, t) := \{j \in \{o_1, \dots, o_{J+1}\} \mid S^o_j \leq t < S^o_j + d_j\} \tag{13}$$

comprises all operations executed at a time $t \geq 0$. Then, we can substitute function $\Phi^{\alpha}_k(t)$ in Formula (4) with

$$r^{\alpha}_k(S^o, t) := \sum_{j \in \mathcal{A}^{\alpha}(S^o, t)} r^{\alpha}_{jk}, \tag{14}$$

which yields the overall amount of a renewable resource $k \in \mathcal{R}^{\alpha}$ required by operations at a time $t \geq 0$. A schedule (S, S^o) to $\mathcal{I}_{\text{RCPSP/c}}$ is *renewable-resource-feasible* if and only if it satisfies availability constraints on renewable resources

$$r^{\alpha}_k(S, t) + r^{\alpha}_k(S^o, t) \leq R^{\alpha}_k \quad (\forall k \in \mathcal{R}^{\alpha}, t \geq 0). \tag{15}$$

For modeling storage facilities within the framework of the RCPSP/c, we consider each storage facility $k \in \mathcal{R}^{\gamma}$ as a *cumulative resource*. The availability of each cumulative resource $k \in \mathcal{R}^{\gamma}$ is constrained by the maximum inventory $R^{\gamma}_k \in \mathbb{Z}_{\geq 0}$. Each operation j moves one material unit from an upstream storage facility to a downstream storage facility or to the sink. To model these moves, we introduce parameters $r^{\gamma}_{jk} \in \{-1, 0, 1\}$ specifying the requirement for a cumulative resource $k \in \mathcal{R}^{\gamma}$ by operation j (cf. Lines 9 and 15 of Algorithm 1). If $r^{\gamma}_{jk} = 1$, operation j *replenishes* one material unit into k at its completion time $(S^o_j + d_j)$ and is called *k-replenishing operation*. If $r^{\gamma}_{jk} = -1$, operation j *depletes* one material unit from k at its start time S^o_j and is called *k-depleting operation*. If $r^{\gamma}_{jk} = 0$, operation j either does not require cumulative resource k or operation j both replenishes and depletes into/from the same cumulative resource k. The latter case will be discussed at the end of this section.

Given S^o, the active set

$$\mathcal{A}^{\gamma+}_k(S^o, t) := \{j \in \{o_1, \dots, o_{J+1}\} \mid r^{\gamma}_{jk} = 1 \ \wedge \ 0 \leq S^o_j + d_j \leq t\} \tag{16}$$

comprises all operations that have replenished material units into cumulative resource $k \in \mathcal{R}^{\gamma}$ and the active set

$$\mathcal{A}^{\gamma-}_k(S^o, t) := \{j \in \{o_1, \dots, o_{J+1}\} \mid r^{\gamma}_{jk} = -1 \ \wedge \ 0 \leq S^o_j \leq t\} \tag{17}$$

comprises all operations that have depleted material units from cumulative resource $k \in \mathcal{R}^{\gamma}$ until a time $t \geq 0$. Then, the overall amount of material units replenished and depleted into/from a cumulative resource $k \in \mathcal{R}^{\gamma}$ by operations at a time $t \geq 0$ is

$$r^{\gamma}_k(S^o, t) := \sum_{j \in \mathcal{A}^{\gamma+}_k(S^o, t)} r^{\gamma}_{jk} + \sum_{j \in \mathcal{A}^{\gamma-}_k(S^o, t)} r^{\gamma}_{jk}. \tag{18}$$

A schedule (S, S^o) to $\mathcal{I}_{\text{RCPSP/c}}$ is *cumulative-resource-feasible* if and only if it satisfies the *maximum inventory constraints*

$$f^{\gamma}_k(S, t) + r^{\gamma}_k(S^o, t) \leq R^{\gamma}_k \quad (\forall k \in \mathcal{R}^{\gamma}, t \geq 0). \tag{19}$$

As introduced in Section 4, $f^{\gamma}_k(S, t)$ is the overall amount of material units released into cumulative resource k at time t. We can now substitute material flow constraints (9d) in the RMCPSP with maximum inventory constraints (19).

Summing up, the RCPSP/c is formulated as follows:

$$\min_{(S, S^o)} \quad z := S_{I+1} \tag{20a}$$

$$\text{subject to} \quad S_{i'} \geq S_i + d_i \quad (\forall (i, i') \in E); \tag{20b}$$

$$S^o_{j'} \geq S^o_j + d_j \quad (\forall (j, j') \in E^{flow}); \tag{20c}$$

$$S^o_j \geq S_i + d^{min}_{ij} \quad (\forall (i, j) \in E^{rel}); \tag{20d}$$

$$r^{\alpha}_k(S, t) + r^{\alpha}_k(S^o, t) \leq R^{\alpha}_k \quad (\forall k \in \mathcal{R}^{\alpha}, t \geq 0); \tag{20e}$$

$$f^{\gamma}_k(S, t) + r^{\gamma}_k(S^o, t) \leq R^{\gamma}_k \quad (\forall k \in \mathcal{R}^{\gamma}, t \geq 0); \tag{20f}$$

$$S_0 = 0; \tag{20g}$$

$$S_i \geq 0 \quad (i = 1, \dots, I+1); \tag{20h}$$

$$S^o_j \geq 0 \quad (j = o_1, \dots, o_{J+1}). \tag{20i}$$

A schedule (S, S^o) to $\mathcal{I}_{\text{RCPSP/c}}$ is *feasible* if and only if it satisfies (20b) – (20i). A time-feasible schedule implicitly satisfies minimum inventory constraints $f^{\gamma}_k(S, t) + r^{\gamma}_k(S^o, t) \geq 0$ $(\forall k \in \mathcal{R}^{\gamma}, t \geq 0)$, since flow-induced precedence constraints (20c) and release constraints (20d) impose that an operation cannot start before a material unit is available. Table 2 summarizes parameters introduced for $\mathcal{I}_{\text{RCPSP/c}}$ in addition to parameters provided in Table 1.

Note that there may be operations $j \in \{o_1, \dots, o_{J+1}\}$ that deplete and replenish from/into the same storage facility, termed *neutral operations* (e.g., operations derived from processing step P1 or P5 in Fig. 1). For each neutral operation j, $r^{\gamma}_{jk} = 0$ holds for each $k \in \mathcal{R}^{\gamma}$. In Algorithm 1, this is implemented in Line 15, where a $(+1)$ is added to $r^{\gamma}_{jk_{step}(w)}$ which can offset a (-1). Although, in reality, each neutral operation reduces the inventory by one material unit during its execution, our model does not consider them when calculating inventories for simplification.

6. Schedule generation scheme

When planning projects with highly granular material flows, we must consider thousands of material units. Given an instance $\mathcal{I}_{\text{RMCPSP}}$ of the problem under study, this section focuses on a procedure for efficiently generating good feasible schedules (S, S^o) to the instance $\mathcal{I}_{\text{RCPSP/c}} := \texttt{Transformation}(\mathcal{I}_{\text{RMCPSP}})$. Each schedule (S, S^o) to $\mathcal{I}_{\text{RCPSP/c}}$ comprises a schedule S to $\mathcal{I}_{\text{RMCPSP}}$.

6.1. Basic concepts

Two well-known techniques for constructing feasible schedules to instances of the classical RCPSP are the serial and parallel SGS. These schemes schedule activities successively at their earliest time- and renewable-resource-feasible start times (cf. Kolisch, 1996b). Priority rule methods are heuristics that combine such a scheme with a priority rule prescribing the order for traversing the activities (cf. Kolisch, 1996a). If maximum time lags between activities exist, the SGS must also allow for unscheduling steps (cf. Franck et al., 2001). Since the availability of cumulative resources depends on all previous resource requirements, both the serial and the parallel SGS do not apply to problems involving such resources straightforwardly. Therefore, exact or heuristic branch-and-bound procedures are frequently employed when cumulative resources are present (cf. e.g., Schwindt & Trautmann, 2000; Neumann et al., 2005). Usually, these procedures start with generating a time-feasible schedule. For each conflict on cumulative resources, i.e., for each violated maximum inventory constraint, additional precedence relations are introduced between activities to resolve the conflict. For each possible set of additional precedence relations that resolves the conflict, a new enumeration node is created, and the steps are repeated until the schedule is cumulative-resource-feasible.

Schwindt et al. (2007) propose a priority rule method using an SGS that can cope with instances of an RCPSP with cumulative

Table 2
Parameters introduced for $\mathcal{I}_{\text{RCPSP/c}}$ in addition to parameters provided in Table 1.

Notation	Denotation
$j = o_1, \ldots, o_{J+1}$	operations; J denotes the number of non-fictitious operations (equals the total number of processing steps applied to material units)
$d_j \in \mathbb{Z}_{\geq 0}$	duration of operation j
$r^{\alpha}_{jk} \in \mathbb{Z}_{\geq 0}$	number of units of renewable resource $k \in \mathcal{R}^{\alpha}$ required by operation j
$(j, j') \in E^{flow}$	flow-induced precedence relation
$(i, j) \in E^{rel}$	release relation
$d^{min}_{ij} \in \mathbb{Z}_{\geq 0}$	minimum time lag between start of activity i and start of operation j
$k \in \mathcal{R}^{\gamma}$	cumulative resource
$r^{\gamma}_{jk} \in \{-1, 0, 1\}$	number of units of cumulative resource $k \in \mathcal{R}^{\gamma}$ required by operation j
$R^{\gamma}_{k} \in \mathbb{Z}_{\geq 0}$	maximum inventory of cumulative resource $k \in \mathcal{R}^{\gamma}$

resources. If maximum inventory constraints for a cumulative resource k are violated in an iteration of the SGS, the latest start times of activities depleting k are reduced sufficiently to resolve the conflict. If this causes infeasibility, activities replenishing k are unscheduled and postponed sufficiently to resolve the conflict. The basic idea behind these shifts is very similar to the introduction of additional precedence relations in branch-and-bound procedures. The schedule generated in each pass corresponds to one single path in the enumeration tree of a branch-and-bound procedure. Schwindt et al. (2007) show that their priority rule method outperforms branch-and-bound procedures for large instances. Therefore, we also opt for a priority rule method to solve the RCPSP/c formulated in Section 5. In contrast to the SGS by Schwindt et al. (2007), our SGS is tailored to the special precedence structure stemming from diverging material flows. We explicitly define how to select operations for scheduling steps. Thereby, we facilitate the search for a feasible solution and accelerate the procedure, as we show in the following.

Let $\overline{Succ}(i) \subseteq \{o_1, \ldots, o_J\}$ denote the set of all non-fictitious operations that are immediate or transitive successors of an activity $i \in \{0, \ldots, I+1\}$. Then, set $M_i := \{i\} \cup \overline{Succ}(i)$ is called a *material flow structure*. If we consider the project as a network, the graph of a material flow structure M_i is a directed outtree that only branches at the root (cf. e.g., $M_1 := \{1, o_1, \ldots, o_{13}\}$ in Fig. 2). This structural property is inherent in diverging material flows, where material units are no longer merged throughout their processing (cf. Section 2). Note that all sets $\overline{Succ}(i)$ are disjoint, i.e., $\overline{Succ}(i) \cap \overline{Succ}(i') = \emptyset$ for all $i, i' \in \{0, \ldots, I+1\}$ with $i \neq i'$. The operations in $\overline{Succ}(i)$ move all material units released by i towards the sink. Thus, in any feasible schedule $(S_i, (S^o_j)_{j \in \overline{Succ}(i)})$ to an RCPSP/c where material flow structure M_i is scheduled isolated from all other activities $i' \neq i$ and operations $j \notin \overline{Succ}(i)$, the inventories of all cumulative resources are zero at time $t = \max_{j \in \overline{Succ}(i)} S^o_j$ (i.e., the time when the last material unit is depleted from a cumulative resource). Schedule $(S_i, (S^o_j)_{j \in \overline{Succ}(i)})$ can be generated straightforwardly: Each branch of the outtree M_i prescribes a linear order on the operations for processing one material unit. If we schedule operations in this order, we get the schedule with the lowest possible load of storage facilities because we always prioritize the subsequent processing of each material unit. That is, if the linear order prescribed by the branch of the outtree does not yield a feasible schedule, the problem is cumulative-resource-infeasible. Appending these linear orders for all branches of the outtree provides a complete linear order on the operations for processing material units released by i. We can efficiently compute an earliest schedule for M_i from a complete linear order as is shown by Carlier et al. (2009). A disjunctive temporal concatenation of feasible schedules $(S_i, (S^o_j)_{j \in \overline{Succ}(i)})$ for all activities $i = 0, \ldots, I+1$ yields a feasible schedule (S, S^o) to $\mathcal{I}_{\text{RCPSP/c}}$. Thus, while a feasible schedule can be found in polynomial time in the case of diverging material flows, this is an NP-complete problem in the general case including converging material flows (Neumann et al., 2003, p. 130). Moreover, in the case of diverging material flows, we already achieve a *good* feasible schedule $(S_i, (S^o_j)_{j \in \overline{Succ}(i)})$ with the described concept due to the low load of storage facilities.

6.2. Decomposition approach

We assume that an instance $\mathcal{I}_{\text{RCPSP/c}}$ typically involves a large number of operations and a smaller number of activities, which is why we decompose the schedule generation by exploiting the special precedence structure of the problem. A feasible schedule (S, S^o) is iteratively constructed in one pass of a serial SGS called `SuperSchedule`, which traverses activities $i = 0, \ldots, I+1$ according to a given priority rule. For each activity i, a separate procedure `SubSchedule` is called, that computes a feasible subschedule $S^M := (S_i, (S^o_j)_{j \in \overline{Succ}(i)})$ for the respective material flow structure M_i, taking into account the current partial schedule (S, S^o) constructed in `SuperSchedule`. However, we schedule M_i without changing other start times $(S_{i'})_{i' \neq i}$ or $(S^o_j)_{j \notin \overline{Succ}(i)}$. In `SuperSchedule`, we do not require unscheduling steps since the inventories of all cumulative resources are zero at time $t = \max_{i=0,\ldots,I+1} \max_{j \in \overline{Succ}(i)} S^o_j$ (i.e., the time when the last material unit is depleted from a cumulative resource) after each iteration. Thus, `SubSchedule` can always compute the feasible subschedule S^M in which activity i starts at the latest completion time in the partial schedule. At this time, no other activity or operation is executed anymore and thus, M_i is isolated (i.e., S^M is feasible, cf. Section 6.1).

Algorithm 2: `SuperSchedule`.

Data: Instance $\mathcal{I}_{\text{RCPSP/c}}$; priority rule π
initialize partial schedule (S, S^o) with $S_i := -\infty$ for $i = 0, \ldots, I+1$ and $S^o_j := -\infty$ for $j = o_1, \ldots, o_{J+1}$;
set $S_0 := 0$;
while *activities i with $S_i = -\infty$ remain* **do**
  determine $\mathcal{E} := \{i \in \{0, \ldots, I+1\} \mid S_i = -\infty \wedge S_{i'} \geq 0 \text{ for each } i' \in Pred(i)\}$;
  select $i^* := \min\{i \in \mathcal{E} \mid \pi(i) = \max_{i' \in \mathcal{E}} \pi(i')\}$;
  compute $S^M := \texttt{SubSchedule}(M_{i^*}, (S, S^o), \mathcal{I}_{\text{RCPSP/c}})$;
  copy values of S^M into partial schedule (S, S^o);
end
set $S^o_{o_{J+1}} := \max_{j=o_1,\ldots,o_J} (S^o_j + d_j)$;
Result: Feasible schedule (S, S^o) to $\mathcal{I}_{\text{RCPSP/c}}$

Algorithm 2 formally describes `SuperSchedule`. It takes priority rule π as input data, where activity i with highest priority has the largest priority value $\pi(i)$. Initially, we set the start times of all activities and operations to $-\infty$, which means that they have not been scheduled yet. As long as unscheduled activities remain, we determine the *eligible set* $\mathcal{E} \subset \{0, \ldots, I+1\}$ that comprises all unscheduled activities i whose immediate predecessors $Pred(i) \subset$

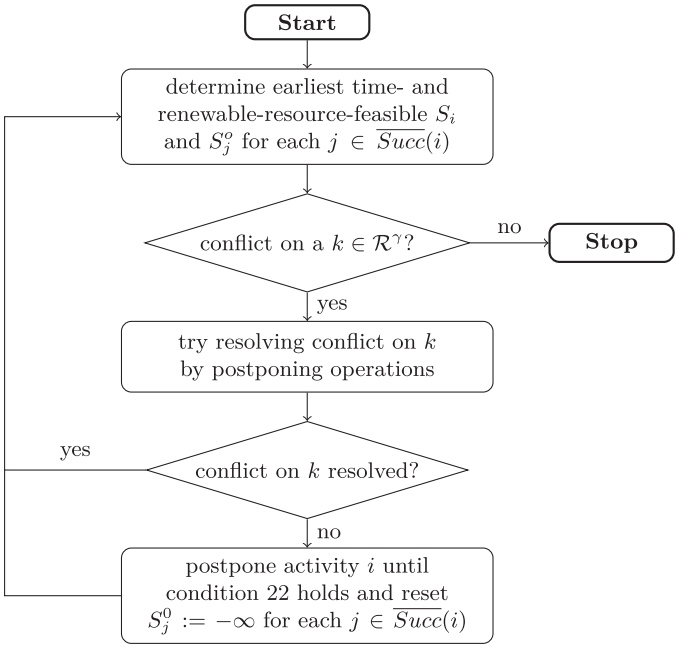

Fig. 3. Flowchart for `SubSchedule` (cf. Algorithm 3).

$\{0, \ldots, I\}$ have been scheduled. We call `SubSchedule` for activity i^* with the highest priority and copy the returned values S^M into the partial schedule (S, S^o). After terminating the while-loop, we schedule fictitious operation o_{J+1}.

Algorithm 3: `SubSchedule`.

Data: Material flow structure M_i; partial schedule (S, S^o); instance $\mathcal{I}_{\text{RCPSP/c}}$

1 sort $\overline{Succ}(i)$ using a depth-first search;
2 **repeat**
3 set S_i to the earliest time- and renewable-resource-feasible start time $t \geq S_i$;
4 set S_j^o to the earliest time- and renewable-resource-feasible start time $t \geq S_j^o$ for each $j \in \overline{Succ}(i)$;
5 **if** *a conflict on a cumulative resource* $k \in \mathcal{R}^\gamma$ *exists* **then**
6 **foreach** t^c *with* $f_k^\gamma(S, t^c) + r_k^\gamma(S^o, t^c) > R_k^\gamma$ **do**
7 construct sets A and B;
8 set $S_j^o := S_{j'}^o - d_j$ for each $(j, j') \in (A \to B)$;
9 **end**
10 **if** *conflict on cumulative resource* k *could not be resolved* **then**
11 increase S_i until condition (22) holds;
12 reset $S_j^o := -\infty$ for each $j \in \overline{Succ}(i)$;
13 **end**
14 **end**
15 **until** *partial schedule* (S, S^o) *is feasible*;

Result: Feasible subschedule $S^M := (S_i, (S_j^o)_{j \in \overline{Succ}(i)})$ to $\mathcal{I}_{\text{RCPSP/c}}$

Algorithm 3 and the flowchart in Fig. 3 formally describe `SubSchedule`. It resolves conflicts on cumulative resources, i.e. violated maximum inventory constraints, preferably by postponing operations. Postponing operations does not directly increase the objective value since operations cannot lie on the critical path. In detail, it works as follows: The material flow structure M_i to be scheduled, the partial schedule (S, S^o), and the instance $\mathcal{I}_{\text{RCPSP/c}}$ are taken as input data. Initially, the set $\overline{Succ}(i)$ is considered as a list, where $\mu(j)$ is the index of an operation j in the list. We order $\overline{Succ}(i)$ so that

(1) $\mu(j') = \mu(j) + 1$ holds for each of the two operations $j, j' \in \overline{Succ}(i)$ if there exists a flow-induced precedence relation $(j, j') \in E^{flow}$, and
(2) $\mu(j) < \mu(j')$ holds for each of the two operations $j, j' \in \overline{Succ}(i)$, if there exist release relations $(i, j), (i, j') \in E^{rel}$ and $d_{ij}^{min} < d_{ij'}^{min}$.

This yields the complete linear order described in Section 6.1. A depth-first search in the outtree M_i can compute the order easily. Then, the earliest time- and renewable-resource-feasible start time $t \geq S_i$ for activity i is chosen as S_i. The same is done with operations in $\overline{Succ}(i)$ that are traversed in the complete linear order.

If the partial schedule (S, S^o) is feasible now, the procedure terminates and returns S^M. Else, we choose one of the cumulative resources $k \in \mathcal{R}^\gamma$ where a conflict exists. In order to keep the project makespan low, we first try to resolve the conflict by postponing k-replenishing operations. If such operations exist in M_i, material units can be buffered in other upstream storage facilities and thus reduce the inventory of k. Otherwise, we must postpone activity i (see below). The set of k-replenishing operations to be postponed is denoted by A. For each time t^c where a conflict exists (i.e., $f_k^\gamma(S, t^c) + r_k^\gamma(S^o, t^c) > R_k^\gamma$), we construct A as an inclusion-maximal subset of $\mathcal{A}_k^{\gamma+}(S^o, t^c) \cap \overline{Succ}(i)$ with

$$f_k^\gamma(S, t^c) + \sum_{j \in \mathcal{A}_k^{\gamma+}(S^o, t^c) \setminus A} r_{jk}^\gamma + \sum_{j \in \mathcal{A}_k^{\gamma-}(S^o, t^c)} r_{jk}^\gamma \geq R_k^\gamma. \tag{21}$$

The active set $\mathcal{A}_k^{\gamma+}(S^o, t^c)$ comprises all scheduled k-replenishing operations that are completed before or at t^c. By using the intersection $\mathcal{A}_k^{\gamma+}(S^o, t^c) \cap \overline{Succ}(i)$, we prevent unscheduling steps for operations that are not part of material flow structure M_i. Condition (21) ensures that we only postpone as many k-replenishing operations as necessary in order to satisfy the maximum inventory constraint for k at time t^c. The left-hand side yields the inventory of k at time t^c after sufficiently postponing all operations $j \in A$. If there are different possibilities for constructing A, we choose k-replenishing operations $j \in \mathcal{A}_k^{\gamma+}(S^o, t^c) \cap \overline{Succ}(i)$ with larger completion times $(S_j^o + d_j)$.

For each operation $j \in A$, we look for a k-depleting operation j' to which j can be assigned. This set of k-depleting operations is denoted by B. We construct B as a subset of $\mathcal{A}_k^{\gamma-}(S^o, \infty) \setminus \mathcal{A}_k^{\gamma-}(S^o, t^c) \equiv \{j' \in \{o_1, \ldots, o_{J+1}\} \mid r_{j'k}^\gamma = -1 \wedge S_{j'}^o > t^c\}$ with $|B| = |A|$. The set $\mathcal{A}_k^{\gamma-}(S^o, \infty) \setminus \mathcal{A}_k^{\gamma-}(S^o, t^c)$ comprises all scheduled k-depleting operations that start after t^c. Since each operation represents the processing of one material unit, condition $|B| = |A|$ ensures a complete assignment of operations in A to operations in B. To improve the performance, we can further exclude those k-depleting operations j' from $\mathcal{A}_k^{\gamma-}(S^o, \infty) \setminus \mathcal{A}_k^{\gamma-}(S^o, t^c)$ whose start time already coincides with a completion time of a corresponding k-replenishing operation j so that the inventory is not reduced (see Appendix A1 in the Supplementary Material for an example). If there are different possibilities for constructing B, we choose k-depleting operations $j' \in \mathcal{A}_k^{\gamma-}(S^o, \infty) \setminus \mathcal{A}_k^{\gamma-}(S^o, t^c)$ with smaller start times $S_{j'}^o$.

Given A and B, we create a bijective relation $A \to B$ assigning each operation $j \in A$ to an operation $j' \in B$. For each $(j, j') \in (A \to B)$, we set $S_j^o + d_j := S_{j'}^o \Leftrightarrow S_j^o := S_{j'}^o - d_j$. In other words, we postpone the completion of k-replenishing operations to times when k-depleting operations start and thus try to reduce the inventory of k at time t^c to the maximum inventory R_k^γ.

The conflict on k is only resolved, if condition (21) is satisfied with equality for each t^c. Then, we repeat with calculating earliest

time- and renewable-resource-feasible start times for activity i and operations $j \in \overline{Succ}(i)$ until partial schedule (S, S^o) is feasible. However, if condition (21) is not satisfied with equality for each t^c, the set $\mathcal{A}_k^{\gamma+}(S^o, t^c) \cap \overline{Succ}(i)$ does not comprise enough k-replenishing operations that could be postponed and therefore we must postpone activity i. We increase S_i until

$$f_k^\gamma(S,t) + r_k^\gamma(S^o,t) \leq R_k^\gamma \quad \left(S_i \leq t < \max_{j \in \overline{Succ}(i)} S_j^o\right) \tag{22}$$

holds. That is, we determine the start time S_i, at which all material units released by activity i fit into k assuming that start times of operations do not change. But because start times of operations can become time- and/or renewable-resource-infeasible due to postponing i, it is not guaranteed that the conflict on k is resolved in one iteration. We must reset $S_j^o := -\infty$ for all $j \in \overline{Succ}(i)$ in the partial schedule which corresponds to an unscheduling. Finally, we repeat by calculating earliest time- and renewable-resource-feasible start times for activity i and operations $j \in \overline{Succ}(i)$. If there are still conflicts on cumulative resources, we further postpone operations or activity i and thus, the procedure will converge towards a feasible subschedule.

We remark that activity i may have to be postponed several times until the partial schedule (S, S^o) is feasible. Besides, the solution space does not necessarily contain the optimal solution since the procedure is not enumerative. We defined, that if there are different possibilities for constructing A, we choose k-replenishing operations $j \in \mathcal{A}_k^{\gamma+}(S^o, t^c) \cap \overline{Succ}(i)$ with larger completion times $(S_j^o + d_j)$. However, it could be better to consider any operation $j \in \mathcal{A}_k^{\gamma+}(S^o, t^c)$ as a possible candidate for A. But such an algorithmic design would result in higher computational effort.

7. Modeling variants

There exist alternative ways for modeling the problem setting described by the RMCPSP and the RCPSP/c. This section introduces these *modeling variants*.

In Section 2, we stated that the work progress of activities is *linear* in time and thus material emerges uniformly distributed over the execution time of activities. Instead, we can assume that work progresses in a *stepwise* fashion and thus, the complete material flow emerges at one point in time for each activity. Assigning the material flow to the start of activities blocks the required storage space in advance and ensures feasibility for any other temporal distribution of emerging material units. In contrast, assigning the material flow to another point in time after the start of activities could lead to infeasible solutions concerning the RMCPSP. Therefore, in a stepwise modeling variant, activities release material flows at their start times.

According to the modeling concepts presented in Section 2, the processing of materials is interruptible down to the level of a single material unit and a discrete time period. This allows the greatest possible planning flexibility. In practical terms, the interruptibility is motivated by the fact that material processing is more similar to mass production than to a project with individual activities. Therefore, we designed the transformation scheme in Section 5 so that a single operation is created for each processing step applied to each material unit. We refer to these single operations as *granular* operations. A large number of granular operations accounts for a huge problem size. We can reduce the problem size by *aggregating* operations, i.e., by replacing several granular operations with one aggregated operation that spans the total duration and the total material flow of the aggregated granular operations. An aggregation of operations is reasonable as long as the processed material units stem from the same activity and follow the same upstream material flow path. Further aggregations would cause an excessive loss of planning flexibility. Aggregated operations can overlap with activities or with each other. The minimum time lags are set to the smallest possible values so that time-feasible start times for aggregated operations are also time-feasible for the corresponding granular operations. Fig. 4 depicts the aggregated variant of the precedence network shown in Fig. 2. For example, operation o'_7 aggregates operations o_7 and o_{10} (both derived from processing step P8). However, it does not aggregate operation o_{12} whose upstream material flow path differs from the others.

To model the original problem as close as possible, we assume that the work progress of aggregated operations and thus, the replenishments and depletions, are linear in time (similar to the continuous flows in Neumann et al. (2005), but with discrete units). However, we can also combine the aggregated and the stepwise modeling variants. That is, replenishments are assigned to the start of operations and depletions to the end of operations. This blocks the required storage space in both the upstream and downstream storage facility.

Computational experiments in Section 8 demonstrate how aggregation affects the solution quality and the performance of solution methods. From a theoretical point of view, the aggregated modeling variant can significantly worsen solutions, as we show with the following example: A project consists of $I = 10$ real activities, where each activity takes ten periods. No renewable resources and precedence relations exist. Each activity releases five material units that traverse material flow path $4 = (\text{S2}, \text{P6}, s)$ (cf. Section 2). The inventory of storage facility S2 is limited to three material units. Processing one material unit in processing step P6 takes one period. In the case of linear work progress, each activity releases one material unit every two periods. The project makespan of an optimal solution of the RMCPSP is ten periods since granular operations can process the material units immediately after each release. Consequently, the inventory of storage facility S2 remains zero and all activities can be started in parallel at time $t = 0$. In contrast, the project takes 55 periods in an optimal solution using the aggregated modeling variant. The aggregated operations, each of which takes five periods, may only start when sufficient material is available so that they can be executed without interruption. Since the last material unit is released at the end of an activity, we must introduce a minimum time lag of six periods between the start of each activity and its subsequent operation. The material units are buffered in S2 until the operations start. Due to the maximum inventory, activities must start offset from each other by five periods, which balances replenishments and depletions. The resulting makespan is $5 \cdot 9 + 10$ (duration of the last activity) $= 55$.

Fig. 5 graphically summarizes the modeling variants. For the original model and each variant, the inventory of storage facility S2 resulting from one activity and its subsequent operation(s) is depicted along with a Gantt chart. The settings described in the above example are illustrated in the left column. The inventory of the storage facility is never lower and/or the material processing never starts earlier than in the original model. Therefore, solutions computed with the modeling variants are also feasible concerning the original RMCPSP. However, this does not apply vice versa. In the case of low maximum inventories, the problem might be infeasible using the modeling variants while the original model would lead to feasible solutions.

8. Computational experiments

In Section 8.1, we sketch a procedure for generating test instances $\mathcal{I}_{\text{RMCPSP}}$ with different characteristics. Section 8.2 explains our experimental design. The results of the experiments are reported in Section 8.3. On this basis, we discuss in Section 8.4,

(1) how the SGS performs in comparison to the constraint programming (CP) solver IBM ILOG CP Optimizer.

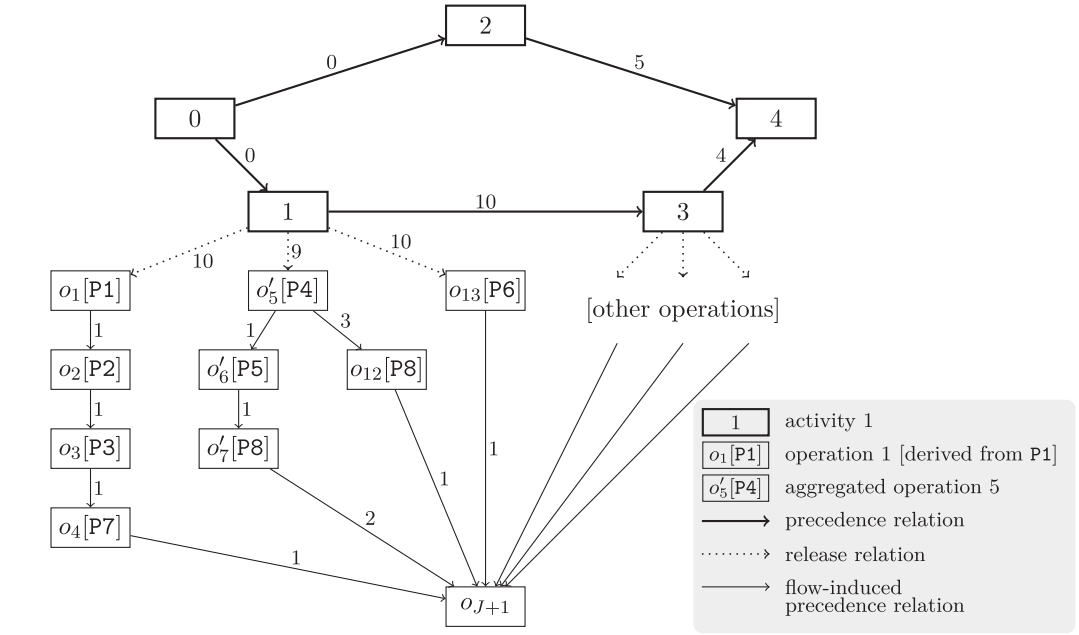

Fig. 4. Exemplary precedence network with aggregated operations (cf. Fig. 2).

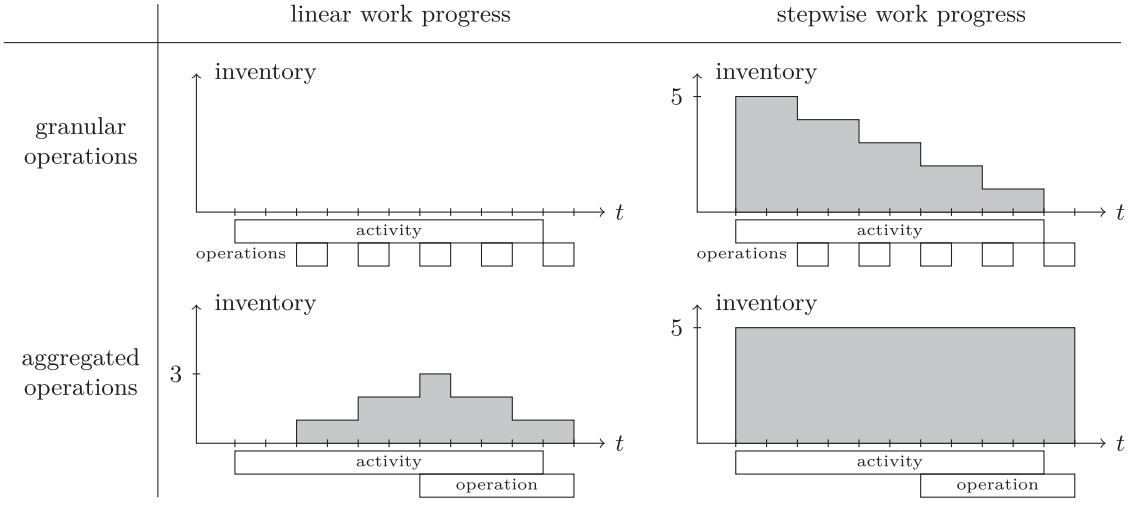

Fig. 5. Inventory diagrams using the original model (linear/granular) and modeling variants.

(2) how the modeling variants impact the solution quality and the performance of the solution methods.
(3) how different characteristics of problem instances impact the performance of the solution methods.
(4) how the problem-specific objective function (9a) impacts the performance of the solution methods.

8.1. Instance generation

To characterize test instances $\mathcal{I}_{\text{RMCPSP}}$, we employ two parameter groups. The first parameter group comprises *project parameters* describing characteristics of project activities and renewable resources as follows:

- $I \in \mathbb{Z}_{>0}$: The number of non-fictitious project activities.
- $NC > 0$: The network complexity, i.e., the average number of non-redundant precedence relations per activity, including the fictitious activities (Kolisch, Sprecher, & Drexl, 1995).
- $RS \in [0, 1]$: The renewable resource strength, which is a measure for the scarceness of renewable resources. The availability of each renewable resource $k \in \mathcal{R}^{\alpha}$ is given by $R_k^{\alpha} := r_k^{\alpha,\min} + RS\,(r_k^{\alpha,\max} + r_k^{\alpha,\min})$, where $r_k^{\alpha,\min}$ ($r_k^{\alpha,\max}$) is the minimal (maximal) requirement of renewable resource k per period (Kolisch et al., 1995).
- $RFA \in [0, 1]$: The renewable resource factor for activities, i.e., the average portion of renewable resources required per activity (Kolisch et al., 1995).

The second parameter group comprises *material flow parameters* that describe characteristics of processing steps, storage facilities, and released material units:

- $INV \in \mathbb{Z}_{\geq 0}$: The maximum inventory of each storage facility.
- $NREL \in \mathbb{Z}_{\geq 0}$: The number of released material units by non-fictitious activities.
- $PREL \in [0, 1]$: The portion of non-fictitious activities releasing material units. More precisely, $\lfloor PREL \cdot I \rfloor = |\{i \in \{1, \ldots, I\} | \sum_{w \in W} f_{iw} > 0\}|$.

- $RFP \in [0, 1]$: The renewable resource factor for processing steps, i.e., the average portion of renewable resources required per processing step.
- $DUR \in \mathbb{R}_{\geq 0}^{|\mathcal{P}|}$: The duration vector for material processing. Each vector component DUR_p specifies the duration for applying processing step $p \in \mathcal{P}$ to one material unit. When transforming $\mathcal{I}_{\text{RMCPSP}}$ into $\mathcal{I}_{\text{RCPSP/c}}$, we set $d_j := DUR_p$ for all operations $j \in O_p$.

To reduce the variability of test instances and to facilitate the comparison between parameter levels, we define that non-fictitious activities either release an equal and deterministic amount of *NREL* or zero material units, i.e., $\sum_{w \in W} f_{iw} \in \{NREL, 0\}$ for each $i = 1, \ldots, I$. Whether the material is released more uniformly or more punctiformly can be controlled via *PREL*. For the same reason, we define that the maximum inventories of all storage facilities are equal and deterministic, i.e., $R_k^{\gamma} := INV$ for each $k \in \mathcal{R}^{\gamma}$. Thus, the availability to store material flows as a whole is controlled via *INV*. The availability to store material flows at individual storage facilities is controlled via *DUR* since the durations of processing steps determine the depletion rates of storage facilities and the depletion rate of a storage facility determines its availability to store material flows.

Instead of generating new instances from scratch, we use existing benchmark instances of the classical RCPSP (single mode and without cumulative resources) from the online library PSPLIB (Kolisch & Sprecher, 1996; 2021) as a starting point. This allows us to directly reuse the best solution found for each PSPLIB instance as a lower bound LB_{PSPLIB} to the objective value of test instances $\mathcal{I}_{\text{RMCPSP}}$. The level of parameter I indicates the set from which we pick the instances. For example, if $I = 30$, we pick instances from the J30-set. In addition to the project parameters listed above, several other parameters describe deterministic or stochastic characteristics of the PSPLIB instances. The majority of them, e.g., the minimum and maximum durations of activities, are fixed and thus, not relevant to our experimental design. For details, we refer to Kolisch & Sprecher (1996).

Each PSPLIB instance comprises a set of four renewable resources $\mathcal{R}^{\alpha} = \{\texttt{R1}, \ldots, \texttt{R4}\}$. The requirements r_{ik}^{α} for $i = 0, \ldots, I + 1$ and $k \in \mathcal{R}^{\alpha}$ lie within the interval $[0, 10]$. With regard to material flows, we define $\mathcal{P} = \{\texttt{P1}, \ldots, \texttt{P8}\}$ as the set of processing steps and $\mathcal{R}_k^{\gamma} = \{\texttt{S1}, \ldots, \texttt{S5}\}$ as the set of storage facilities (cf. Fig. 1). Initially, processing steps do not require resources $k \in \{\texttt{R1}, \ldots, \texttt{R4}\}$. For modeling the constrained processing capacity, we introduce additional renewable resources, called *machines* (to distinguish them from PSPLIB resources). We define that both processing steps P3 and P4 require one unit of a shared machine MA1 and both processing steps P6 and P7 require one unit of MA2. Processing step P1 requires one unit of an individual machine MA3, P2 one unit of MA4, P5 one unit of MA5, and P8 one unit of MA6. We set the maximum availability $R_k^{\alpha} := 1$ for $k \in \{\texttt{MA1}, \ldots, \texttt{MA6}\}$ and the set of renewable resources $\mathcal{R}^{\alpha} := \{\texttt{R1}, \ldots, \texttt{R4}\} \cup \{\texttt{MA1}, \ldots, \texttt{MA6}\}$. Project activities do not require machines. Thus, machines represent a fixed, constrained processing capacity, so that bottlenecks for material flows are only controlled via *INV* and *DUR*. We randomly choose $\lfloor RFP \cdot |\mathcal{P}| \cdot 4 \rfloor$ tuples (p, k) out of all combinations of processing steps $\mathcal{P}$ and resources $\{\texttt{R1}, \ldots, \texttt{R4}\}$. For each chosen tuple (p, k), we draw the resource requirement r_{pk}^{α} randomly out of [1,3]. Thus, parameter *RFP* determines the extent to which activities and processing steps compete against renewable resources R1, ..., R4. By using the interval [1,3] for r_{pk}^{α}, we make sure that the processing of possibly thousands of material units does not dominate the requirements of the four renewable resources.

In the last step, we randomly select $\lfloor PREL \cdot I \rfloor$ activities. For each selected activity i, we simulate *NREL* random material flow paths through the network depicted in Fig. 1. The number of released material units f_{iw} is set to the amount a path $w \in W$ has been simulated for each activity i.

8.2. Experimental design

Since we mainly focus on the impact of material flow constraints, we fix the project parameters *NC* to 2.1, *RS* to 0.5, and *RFA* to 0.5. Results from experiments with variable project parameters *NC*, *RS*, and *RFA* are presented in Appendix A2.

The first two columns of Table 3 show the different parameter levels we used in a full factorial experimental design. Consequently, we get a total of $2^5 \cdot 3 = 96$ test instances. The number of operations per instance varies between 1,073 and 76,257. By aggregating operations, these numbers reduce considerably to 106 and 2,348. The levels for *DUR* are the vectors $dur_1 = (0.06, 0.06, \ldots, 0.06)$, $dur_2 = (0.1, 0.08, 0.06, 0.1, 0.04, 0.1, 0.02, 0.02)$, and $dur_3 = (0.02, 0.04, 0.06, 0.02, 0.08, 0.04, 0.1, 0.08)$. These values were chosen so that the expected durations for completely processing one material unit are approximately equal (≈ 0.2). With dur_1, all processing steps require the same time. Vector dur_2 describes a situation where the bottleneck is in the upstream part of the material flows. With dur_3, it is in the downstream part, respectively.

We transformed $\mathcal{I}_{\text{RMCPSP}}$ into $\mathcal{I}_{\text{RCPSP/c}}$ and employed both the SGS and CP techniques alternatively. As their performance has improved in the past years, CP solvers increasingly become a relevant tool to solve scheduling problems (e.g., Gerhards, 2020). We chose IBM ILOG CP Optimizer in IBM ILOG CPLEX Optimization Studio 12.9.0 for our study, since it is robust and quickly accessible via the IBM ILOG Concert Technology library (Laborie, 2009; Laborie, Rogerie, Shaw, & Vilím, 2018). Although the library is extensive, it possesses some limitations that constrain the modeling scope. Function `stepAtStart` (`stepAtEnd`) models a requirement of a cumulative resource at the start (at the end) of an activity or operation. However, modeling material flows that emerge uniformly distributed over the execution time of an activity (cf. Formula (7)) is not possible. Therefore, when using CP Optimizer, we must restrict ourselves to the stepwise modeling variant. Apart from that, the Concert Technology library offers all functions for equivalently expressing the RCPSP/c constraints (20b)–(20e) and (20g)–(20i) as well as the objective as a CP formulation (see Appendix A3).

We tested all possible combinations of solution methods (CP and SGS) and models (original model and modeling variants). For the SGS, we distinguish between the results from a single-pass configuration using priority rule "latest start time (LST)" (cf. Kolisch, 1996b) and a multi-pass configuration. In our multi-pass configuration, the SGS schedules activities in the order specified by a precedence feasible activity list. After each SGS pass, a portion ρ is drawn randomly out of $[0, 0.4)$. The activity list that led to the best solution so far is duplicated and $\rho \cdot I$ random activities are shifted to random precedence-feasible positions. This is repeated until a time limit is reached. For CP Optimizer, we began with the default cold start (CS) configuration, i.e., without providing any information about known solutions. Since we can quickly generate a feasible solution with the SGS, we also tried a warm start (WS) configuration setting the LST solution as the starting point for CP Optimizer. A combination of solution method, model, and configuration (e.g., SGS/linear/granular/multi) is referred to as *solution strategy*.

Unless otherwise stated, a time limit of ten minutes per instance was used for CP Optimizer and the multi-pass SGS. We consider this a practicable runtime for comparing the different solution strategies. To assess further potential for optimization, we additionally tested the warm configurations of CP Optimizer with a time limit of 60 minutes.

Table 3
Relative deviation from the tightest lower bound [%].

Parameters	Levels	SGS								CP			
		linear				stepwise				stepwise			
		granular		aggregated		granular		aggregated		granular		aggregated	
		LST	multi	LST	multi	LST	multi	LST	multi	CS	WS	CS	WS
Overall		14.02	9.11	14.46	8.98	14.72	9.83	17.94	11.34	2.91[a]	7.37[a]	**5.28**	5.30
I	30	20.39	12.83	21.11	12.25	21.34	13.71	25.57	15.54	3.07[a]	8.33	**4.58**	4.65
	120	7.65	**5.39**	7.81	5.70	8.10	5.94	10.30	7.15	2.65[a]	6.32[a]	5.98	5.94
INV	1000	12.91	7.31	13.58	7.15	13.46	7.98	15.64	8.38	1.88[a]	3.77[a]	2.40	**2.35**
	200	15.13	10.91	15.34	10.80	15.98	11.68	20.24	14.30	5.10[a]	10.97[a]	**8.16**	8.24
NREL	50	12.87	7.49	13.14	6.92	13.10	7.73	13.89	7.61	2.22[a]	4.36	**2.29**	2.30
	200	15.17	10.73	15.77	11.04	16.35	11.93	21.98	15.07	4.65[a]	10.65[a]	**8.27**	8.29
PREL	0.25	13.42	7.51	13.81	7.50	14.10	8.26	17.34	10.11	1.58[a]	5.58	**4.54**	**4.54**
	1	14.62	10.71	15.11	10.45	15.34	11.39	18.53	12.57	5.50[a]	9.32[a]	**6.03**	6.05
RFP	0	5.64	**3.97**	6.13	4.36	6.45	4.83	9.63	6.88	1.24[a]	5.32	5.26	5.29
	0.5	22.40	14.25	22.79	13.59	22.99	14.83	26.24	15.80	4.52[a]	9.61[a]	**5.30**	**5.30**
DUR	dur_1	10.74	6.37	10.81	5.94	11.38	7.30	15.09	8.73	4.42[a]	4.11	**2.63**	2.68
	dur_2	16.6	10.45	16.30	10.26	17.39	11.12	19.22	12.31	3.08[a]	8.30[a]	**6.98**	7.00
	dur_3	14.72	10.51	16.27	10.73	15.39	11.06	19.49	12.98	0.00[a]	9.91[a]	6.24	**6.20**

[a] CP Optimizer did not find a feasible solution for all instances.

Table 4
Proportion of instances for which the best known solution was found [%].

Parameters	Levels	SGS								CP			
		linear				stepwise				stepwise			
		granular		aggregated		granular		aggregated		granular		aggregated	
		LST	multi	LST	multi	LST	multi	LST	multi	CS	WS	CS	WS
Overall		7.29	**61.46**	6.25	31.25	7.29	25.00	6.25	27.08	38.54	41.67	47.92	51.04
I	30	0.00	56.25	0.00	33.33	0.00	29.17	0.00	27.08	52.08	56.25	**64.58**	62.50
	120	14.58	**66.67**	12.50	29.17	14.58	20.83	12.50	27.08	25.00	27.08	31.25	39.58
INV	1000	8.33	**62.50**	6.25	35.42	8.33	31.25	6.25	33.33	50.00	52.08	56.25	**62.50**
	200	6.25	**60.42**	6.25	27.08	6.25	18.75	6.25	20.83	27.08	31.25	39.58	39.58
NREL	50	14.58	62.50	12.50	50.00	14.58	41.67	12.50	47.92	64.58	68.75	**75.00**	66.67
	200	0.00	**60.42**	0.00	12.50	0.00	8.33	0.00	6.25	12.50	14.58	20.83	35.42
PREL	0.25	12.50	60.42	12.50	47.92	12.50	35.42	12.50	41.67	62.50	64.58	66.67	**70.83**
	1	2.08	**62.50**	0.00	14.58	2.08	14.58	0.00	12.50	14.58	18.75	29.17	31.25
RFP	0	14.58	**85.42**	12.50	47.92	14.58	43.75	12.50	41.67	39.58	45.83	43.75	50.00
	0.5	0.00	37.50	0.00	14.58	0.00	6.25	0.00	12.50	37.50	37.50	**52.08**	**52.08**
DUR	dur_1	6.25	**59.38**	6.25	31.25	6.25	25.00	6.25	21.88	40.63	43.75	40.63	46.88
	dur_2	6.25	53.13	6.25	28.13	6.25	21.88	6.25	25.00	37.50	37.50	53.13	**56.25**
	dur_3	9.38	**71.88**	6.25	34.38	9.38	28.13	6.25	34.38	37.50	43.75	50.00	50.00

We carried out all experiments on an AMD Ryzen 9 (4.0 gigahertz, 12 cores) with 128 gigabyte RAM. Both the transformation scheme and the SGS were coded in Java and compiled with Eclipse 2019–12. For CP Optimizer, we used the automatic search and the default parameter levels of CP Optimizer since IBM advertises a "model-and-run development process" (Laborie (2009)).

8.3. Results

We present the results of the experiments in Tables 3–5, using a specific measure for each table. In the tables' rows, we distinguish between the different parameter levels and average the observed values over all levels except for the one under consideration. The values of the best-performing solution strategy are set in bold.

Table 3 shows the relative deviations from the tightest of two lower bounds, namely:

(1) The best solution found for the respective PSPLIB instance, denoted by LB_{PSPLIB} (provided in Kolisch & Sprecher, 2021).
(2) A material flow-based lower bound, denoted by LB_{flow}. It is the minimum time required for released material units passing through the material flow network's tightest bottleneck. We provide more details in Appendix A4.

The results of columns CP/granular must be interpreted carefully since these solution strategies could not find a feasible solution for all instances: Using a cold start (warm start), 53 (92) out of 96 test instances were solved to feasibility. Table 3 only depicts the deviations averaged over these solved instances. We observed more extreme deviations with the SGS than with CP Optimizer. For example, the deviation is larger than 30% for 11 instances using SGS/linear/granular/multi compared to one instance using CP/aggregated/CS. It is well known that a mean is influenced by such outliers. Therefore, Table 4 provides the proportion of instances, for which a solution strategy found the best known objective value. This measure also includes unsolved instances.

Table 5 provides insight into the computation times until the first feasible solution was found. The multi-pass configurations of the SGS are excluded since their first pass was always an LST pass.

Apart from the results depicted in the tables, we made the following observations:

- Within ten minutes, the multi-pass configurations were able to perform an average of 29,034 SGS passes per instance for SGS/linear/granular, 32,313 for SGS/stepwise/granular, 1,245,726 for SGS/linear/aggregated and 1,423,251 for SGS/stepwise/aggregated.

Table 5
Computation times until a feasible solution was found [s].

Parameters	Levels	SGS[b]				CP			
		linear		stepwise		stepwise			
		granular	aggregated	granular	aggregated	granular		aggregated	
		LST	LST	LST	LST	CS	WS	CS	WS
Overall		3.28	0.13	3.86	**0.09**	44.40[a]	54.25[a]	1.51	0.24
I	30	0.17	0.01	0.13	**0.01**	32.62[a]	6.92	0.27	0.12
	120	6.39	0.25	7.60	**0.18**	63.83[a]	105.88[a]	2.75	0.35
INV	1000	1.42	0.14	1.13	**0.09**	63.78[a]	54.17[a]	1.44	0.22
	200	5.14	0.12	6.59	**0.09**	3.35[a]	54.33[a]	1.58	0.25
NREL	50	0.32	0.03	0.25	**0.02**	18.04[a]	8.01	0.97	0.22
	200	6.24	0.23	7.48	**0.16**	111.16[a]	104.70[a]	2.05	0.25
PREL	0.25	0.26	0.01	0.20	**0.01**	20.72[a]	8.77	0.23	0.12
	1	6.30	0.25	7.52	**0.17**	90.43[a]	103.87[a]	2.79	0.35
RFP	0	3.38	0.15	3.45	**0.11**	41.25[a]	70.88	1.38	0.21
	0.5	3.17	0.11	4.27	**0.07**	47.43[a]	36.11[a]	1.64	0.26
DUR	dur_1	3.43	0.04	3.12	**0.02**	52.69[a]	66.00	1.17	0.24
	dur_2	4.34	0.04	4.24	**0.02**	60.72[a]	50.24[a]	1.21	0.24
	dur_3	2.07	0.31	4.23	0.24	2.67[a]	45.74[a]	2.15	**0.23**

[a] CP Optimizer did not find a feasible solution for all instances.
[b] Computation times averaged over ten repetitions per instance.

- CP/granular could prove optimality for 31 (34) instances using the cold start (warm start) configuration. CP/aggregated could prove optimality for 38 instances using both the cold start or the warm start configuration. These are all instances, for which the computed objective value equals LB_{PSPLIB}.
- The 43 instances for which CP/granular/CS could not find a feasible solution are almost all instances with $(INV, NREL) = (200, 200)$, i.e., low maximum inventories in relation to the total number of released material units. But also others, e.g., two instances with $(INV, NREL) = (1000, 50)$, are unsolved.
- The four instances for which CP/granular/WS could not find a feasible solution comprise more than 75,000 operations each, i.e., they belong to the largest instances investigated.
- By increasing the time limit to 60 minutes per instance, CP/granular/WS (CP/aggregated/WS) was able to improve the solution of 30 (26) instances compared to the solutions found after ten minutes. The achieved reductions of the objective values range from 0.01 (0.01) to 18.96% (1.32%). The overall mean deviation from the tightest lower bound is now 6.44% (5.00%) in contrast to 7.37% (5.30%) after ten minutes. For the six instances with more than 75,000 operations and $(INV, NREL) = (200, 200)$, CP/granular/WS crashed due to a lack of memory.

We repeated all computational experiments for the problem with objective function $z' := \max\{S_{l+1}, S^o_{0_{j+1}}\}$ instead of $z := S_{l+1}$. That is, minimizing the makespan of the project and the material processing instead of only minimizing the project's makespan. This does not treat material flows on a secondary level anymore and corresponds to the makespan-minimization objective typically studied in project scheduling problems. The results can be found in Appendix A5.

Additionally, we employed the multi-start configuration in combination with the SGS proposed by Schwindt et al. (2007) (cf. Section 6.1). This SGS is referred to as *generic SGS* since it does not distinguish between activities and operations, but constructs a schedule regardless of the material flow structure. Thus, unscheduling steps are required to resolve conflicts on cumulative resources. In the case of granular (aggregated) operations, the generic SGS found a feasible solution for 37 (66) out of the 96 test instances within the ten-minute time limit. For the remaining instances, all SGS passes failed due to infinite unscheduling loops. Detailed results are presented in Appendix A6.

8.4. Discussion

8.4.1. Impact of the solution method

The deviations from the tightest lower bound suggest that CP/granular/CS computes the best solutions with an overall mean deviation of 2.91%. However, this mean only covers the 53 instances for which CP/granular/CS found a feasible solution. Using a warm start configuration helped to find feasible solutions for 92 instances. To our surprise, there are still four unsolved instances, although CP Optimizer got a feasible starting solution. We suspect that the engine does not manage to process the starting solutions within the ten-minute time limit. The search log does not shed light on the reasons but only says "Using starting point solution" and "Search terminated by limit, no solution found". Using the warm start, the overall mean deviation from the tightest lower bound is significantly higher (7.37%) compared to the cold start. This is because harder instances have also been solved and are included here. Thus, the warm start results of CP/granular are more meaningful than the cold start results. But due to the remaining unsolved instances, they are still not fully comparable to other values in Table 3. Allowing a runtime of 60 minutes, CP/granular/WS can considerably improve some solutions so that the mean deviation comes up to 6.44%.

With an overall mean deviation of at most 5.30%, we identify CP/aggregated as the best-performing solution strategy on average. This remains true, if we permit a runtime of 60 minutes per instance, which results in a mean deviation of 5.00% using a warm start. Overall, the cold start and warm start configuration of CP/aggregated roughly perform equal in terms of deviations from the tightest lower bound. Comparing instance by instance, we find that for 16 instances, CP/aggregated/WS computed strictly better solutions than CP/aggregated/CS. Vice versa, for 21 instances CP/aggregated/CS did strictly better. This indicates that CP Optimizer can get distracted by a starting solution.

With regard to the SGS, the best overall deviation is 8.98% for SGS/linear/aggregated/multi. As expected, the multi-pass configuration can significantly improve the initial LST solutions for all investigated models.

In Table 3, the lower bounds represent a common reference point for measuring the solution quality numerically. Table 4 compares the solution strategies among each other and uses a binary measure of the type "best known solution or not best known solution". Here, we clearly observe that the multi-pass SGS in combi-

nation with the original model (i.e., SGS/linear/granular/multi) performs best. For 61.46% of the 96 test instances, it found the best known solution. CP Optimizer could find the best known solution for 51.04% of the instances using aggregated operations and a warm start.

Table 5 shows that the computation times until a feasible solution was found differ between both solution methods (particularly in the case of granular operations). Again, the measures for CP/granular must be interpreted carefully due to unsolved instances. It becomes obvious that the SGS finds feasible solutions very quickly since it can exploit the problem-specific precedence structure. Providing a starting solution helps CP Optimizer to reduce the effort for finding an initial solution as the values for CP/aggregated prove.

Since the results in Tables 3 and 4 are dichotomous, it is difficult to judge one solution method as superior to the other. CP Optimizer outperforms the SGS as long as we measure the solution quality numerically. However, if we use the binary measure, the overall result turns the other way round. For 40 instances, the SGS found a strictly better solution than any of the CP strategies. And for 27 instances, both solution methods computed the same objective value. For the remaining 29 instances, only CP Optimizer came up with the best known solution. Recall that the SGS can solve the original RMCPSP with linear work progress whereas CP Optimizer solved a variant of the problem, i.e., the comparison is somewhat distorted. Thus, if material flows emerge uniformly in a practical problem setting, we recommend to apply both solution methods.

Finally, we briefly compare our SGS with the generic SGS by Schwindt et al. (2007). Using the generic SGS, we found at most 37.50% of the best known solutions (cf. Appendix A6). That is, the generic SGS performs worse overall than the problem-specific SGS proposed in this paper. Instances with granular operations were solved by the generic SGS only if maximum inventory constraints do not bind. Also in the case of aggregated operations, low maximum inventories combined with many released materials pose a hard challenge for the generic SGS. On the other hand, for 17 out of the 96 instances, the generic SGS found strictly better solutions than any solution strategy using our problem-specific SGS. In these cases, the decomposition approach seems to hinder the search of the solution space.

8.4.2. Impact of the model

The aggregation has reduced the number of operations per test instance by 93% on average. But aggregation is accompanied by a loss of modeling accuracy since we discard the interruptibility of material flow processing and, thus, the full flexibility to replenish and deplete storage facilities (cf. Section 7). According to Table 4, the aggregation can considerably lower the solution quality obtained with the SGS. While SGS/linear/granular/multi accounts for 61.46% of the best known solutions, it is only 31.25% for the respective aggregated variant. Comparing the two columns SGS/linear/granular/LST and SGS/linear/aggregated/LST in Table 3 sheds light on the extent to which solutions are worsened by the aggregation. Using the same priority rule, the average deviation from the tightest lower bound increases by 0.44% due to the aggregation. This is by far less significant than in the theoretical example described in Section 7. While there was only one storage facility in the theoretical example, the material flows distribute to five storage facilities in the test instances. Besides, the theoretical example does not comprise renewable resources that constrain the execution of material processing. In the test instances, however, renewable resource constraints delay the material processing anyway such that materials must be buffered in storage facilities. The storage load enforced by minimum time lags between the activities and aggregated operations thus becomes less relevant than in the theoretical example. In general, it can be concluded that the impact of an aggregation strongly depends on the characteristics of an instance. The more similar the material flows are and the fewer renewable resource constraints exist, the higher the risk of not adequately representing a problem setting with aggregated operations. Contrarily, we observe that the aggregation may also have a positive effect on the solution quality. If we measure the deviation from the tightest lower bound as done in Table 3, we find that, on average, SGS/linear/aggregated/multi performs better than SGS/linear/granular/multi. This observation can be explained by the fact that within ten minutes, the multi-pass procedure was able to perform an average of 1,245,726 SGS passes per instance using aggregated operations instead of only 29,034 SGS passes using granular operations. Thus, a much larger part of the solution space was searched. The computation times in Table 5 confirm this effect. With regard to CP Optimizer, it is evident that smaller instance sizes accelerate its search considerably. All instances with aggregated operations could be solved to feasibility within 0.06 to 10.15 seconds using the cold start configuration. CP/aggregated could also find more best known solutions than CP/granular.

Switching from linear work progress to stepwise work progress blocks storage space and thus, goes along with a higher inventory in storage facilities (cf. Fig. 5). As expected, this affects the solution quality. If we compare the two columns SGS/linear/granular/LST and SGS/stepwise/granular/LST in Table 3, we find that the average relative deviation from the tightest lower bound increases by 0.70% using the stepwise modeling variant. After running the multi-pass procedure, this deterioration is 0.72%, i.e., remains almost unchanged. Similarly, the multi-pass configurations could find less best known solutions after switching to the stepwise variant. Due to this significant effect, it can be assumed that CP Optimizer would perform even better if it could be applied to the linear modeling variant. Indeed, if we compare just the results of the stepwise modeling variant, CP Optimizer clearly outperforms the SGS.

8.4.3. Impact of the parameters

The most obvious impact on the performance of both solution methods is related to the parameters I and RFP. For $I = 30$, CP Optimizer's solutions deviate 4.58% from the tightest lower bound after running its best solution strategy CP/aggregated/CS (cf. Table 3). Using the SGS, the lowest averaged deviation is 12.25%. For $I = 120$, these measures change to 5.94% for CP Optimizer and 5.39% for the SGS, i.e., the SGS now performs better than CP Optimizer. We recognize the same effect in Table 4 (64.58% vs. 56.25% for $I = 30$ and 39.58% vs. 66.67% for $I = 120$). Hence, while the SGS benefits from additional activities in the instances, the performance of CP Optimizer suffers. We suspect the reason for this to be the higher number of released material units, which is a consequence of the higher number of activities. For a deeper investigation, we created a scatter plot showing the deviations from the tightest lower bound against the number of released material units for the overall best solution strategy of each solution method (cf. Fig. 6). Resulting from the possible combinations of parameter levels, the activities release a total of 350, 1,400, 1,500, 6,000, or 24,000 material units per instance. The scatter plots reveal that the SGS performs poorly for the instances with a low number of released material units compared to CP Optimizer. However, it performs better for instances with 6,000 units. This can be explained as follows: For a low number of released material units, the availability constraints on renewable resources bind more tightly than maximum inventory constraints. For a large number of released material units, this relation swaps. And since the decomposition concept of the SGS is tailored to the maximum inventory constraints, it performs better in the latter case. In contrast, CP Optimizer handles renewable resource very well. This also explains the significant influence of the RFP parameter. If $RFP = 0$, activities and operations do not compete for renewable resources. Hence, the execution of operations is

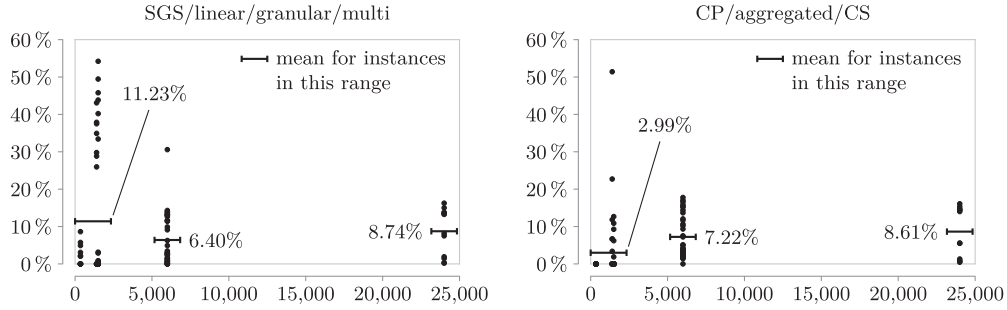

Fig. 6. Scatter plots showing the deviation from the tightest lower bound (y-axis) against the number of released material units (x-axis) for each instance.

less constrained by the availability of renewable resources and the SGS performs better. For $RFP = 0.5$, however, CP Optimizer clearly outperforms all SGS solution strategies.

For instances with a total of 24,000 released material units, Fig. 6 indicates that the SGS and CP Optimizer perform almost equally. Here, all activities release large amounts of material units, so that there is less space for improving a feasible solution by rearranging the activities. Both solution methods could only slightly improve their initial solutions (0.34% for the SGS and 4.15% for CP Optimizer on average). It can be assumed that any solution method will come up with similar results.

Regarding the influence of parameter *DUR* in Table 4, we find that CP Optimizer computes the best solutions for instances with bottlenecks in the upstream part of the material flows ($DUR = dur_2$). Conversely, instances with bottlenecks in the downstream part ($DUR = dur_3$) are better solved by the SGS. We suppose that the latter pose a harder challenge for CP Optimizer since downstream conflicts are only fed back to the activities' level after several backsteps.

Finally, it is not surprising that CP/granular/CS struggles with solving the $(INV, NREL) = (200, 200)$ instances. Since maximum inventories are low in relation to the number of released material units, maximum inventory constraints bind tightly. The SGS, however, proves that all test instances are feasible.

8.4.4. Impact of the objective function

The objective function (9a) of the RMCPSP was chosen since the project planning and the material flow processing are organizationally separated from each other in many real-world settings. In contrast to the usual makespan minimization in the literature, we do not minimize the makespan of the complete schedule, but only of the project activities. This is a key feature of diverging material flows studied in this paper. Nevertheless, all concepts introduced can be transferred to the case of minimizing the total makespan $z' := \max\{S_{l+1}, S^o_{0_{J+1}}\}$ instead of $z := S_{l+1}$. For the SGS, we simply change the measure for evaluating the quality of a solution. For CP Optimizer, we exchange the objective in our problem formulation.

According to the results presented in Appendix A5, SGS/stepwise/granular/multi performs best with z'. Overall, it even outperforms CP Optimizer in terms of average deviations from the tightest lower bound and proportion of instances solved to the best known objective value. However, the ranking differs for the individual parameter levels so that we cannot identify a clear superiority of any solution strategy.

The most interesting difference between both objective functions is that CP/aggregated could prove optimality for 92 instances with z' compared to 38 instances with the original objective z. It seems that our atypical objective z makes it more difficult to identify a solution as being optimal. Using the information about optimality, we made the following observation with z': For 47 instances, the SGS computed a better solution than CP/aggregated although CP/aggregated proved optimality. Hence, one should be aware that the optimal solution the problem formulated with modeling variants is not necessarily the optimal solution concerning the original model.

9. Summary and conclusions

This paper introduces and discusses a new project scheduling problem termed the *resource-and-material-flow-constrained project scheduling problem (RMCPSP)*. The problem incorporates finish-to-start precedence constraints between activities, maximum availability constraints on renewable resources, and material flow constraints. The latter arise from bottlenecks in processing and storing material units that are released by project activities. Material flows are dealt with on a secondary planning level and indirectly impact the project planning in the case of congestion in the material flow network. The RMCPSP is the first project scheduling problem suitable for planning projects dealing with diverging material flows, such as dismantling projects.

To tackle the problem, we transformed instances of the RMCPSP into instances of a resource-constrained project scheduling problem with cumulative resources. The transformation allows solving the problem with well-known strategies from the literature, such as priority rule methods, which we have chosen for our approach. To this end, we presented a schedule generation scheme (SGS) exploiting outtree-like precedence structures in the problem. We decomposed the SGS into individual scheduling passes for so-called material flow structures, each of which consists of a single activity and all its released material flows. The SGS is particularly useful for understanding structural properties, judging the feasibility of problem instances, and computing starting solutions. Computational experiments show that IBM ILOG CP Optimizer is another promising tool to solve the problem.

Comparing the results of the experiments, we made several findings that help to adequately model a real-world problem: Whether materials are released and processed uniformly distributed (similar to so-called continuous material flows, but with discrete units) or in a stepwise fashion has a significant impact on the solution quality. Taking this into account, a decision-maker should carefully choose the desired modeling concept. Studying this impact with CP Optimizer was not possible due to a lack of appropriate programming interfaces. Whether material processing is modeled as interruptible or not affects the solution quality as well. Enabling interruptibility means more flexibility to utilize the full storage capacity which can be relevant in crowded dismantling sites. However, a solver's performance can suffer as the experiments with CP Optimizer reveal. Giving up interruptibility to some extent is found to be beneficial for the solution quality. This trade-off should be considered when determining the number of separately scheduled entities.

Beyond the contents of this paper, other typical concepts of project scheduling could be integrated into the RMCPSP, such as multiple execution modes, nonrenewable resources, general temporal relations, and shelf-life times. The integration of the SGS

into more sophisticated metaheuristics such as simulated annealing could be a promising way to improve the solution quality. When using CP Optimizer, one could attempt to better imitate uniformly distributed material flows by splitting activities and linking them via no-wait precedence relations. Finding the best relation between the size of such instances and a model that is close to reality would require an extensive series of experiments.

Declaration of Competing Interest

None.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2022.03.047.

References

Bartusch, M., Möhring, R. H., & Radermacher, F. J. (1988). Scheduling project networks with resource constraints and time windows. *Annals of Operations Research, 16*(1), 199–240. https://doi.org/10.1007/BF02283745.

Baumann, P., & Trautmann, N. (2014). A hybrid method for large-scale short-term scheduling of make-and-pack production processes. *European Journal of Operational Research, 236*(2), 718–735. https://doi.org/10.1016/j.ejor.2013.12.040.

Belaid, R., T'kindt, V., & Esswein, C. (2012). Scheduling batches in flowshop with limited buffers in the shampoo industry. *European Journal of Operational Research, 223*(2), 560–572. https://doi.org/10.1016/j.ejor.2012.06.035.

Blazewicz, J., Lenstra, J. K., & Kan, A. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics, 5*(1), 11–24. https://doi.org/10.1016/0166-218X(83)90012-4.

Blömer, F., & Günther, H.-O. (1998). Scheduling of a multi-product batch process in the chemical industry. *Computers in Industry, 36*(3), 245–259. https://doi.org/10.1016/S0166-3615(98)00075-X.

Blömer, F., & Günther, H.-O. (2000). LP-based heuristics for scheduling chemical batch processes. *International Journal of Production Research, 38*(5), 1029–1051. https://doi.org/10.1080/002075400189004.

Boysen, N., Bock, S., & Fliedner, M. (2013). Scheduling of inventory releasing jobs to satisfy time-varying demand: An analysis of complexity. *Journal of Scheduling, 16*(2), 185–198. https://doi.org/10.1007/s10951-012-0266-0.

Briskorn, D., Choi, B.-C., Lee, K., Leung, J., & Pinedo, M. (2010). Complexity of single machine scheduling subject to nonnegative inventory constraints. *European Journal of Operational Research, 207*(2), 605–619. https://doi.org/10.1016/j.ejor.2010.05.036.

Briskorn, D., Jaehn, F., & Pesch, E. (2013). Exact algorithms for inventory constrained scheduling on a single machine. *Journal of Scheduling, 16*(1), 105–115. https://doi.org/10.1007/s10951-011-0261-x.

Briskorn, D., & Leung, J. Y.-T. (2013). Minimizing maximum lateness of jobs in inventory constrained scheduling. *Journal of the Operational Research Society, 64*(12), 1851–1864. https://doi.org/10.1057/jors.2012.155.

Briskorn, D., & Pesch, E. (2013). Variable very large neighbourhood algorithms for truck sequencing at transshipment terminals. *International Journal of Production Research, 51*(23–24), 7140–7155. https://doi.org/10.1080/00207543.2013.849825.

Briskorn, D., & Zeise, P. (2019). A cyclic production scheme for the synchronized and integrated two-level lot-sizing and scheduling problem with no-wait restrictions and stochastic demand. *OR Spectrum, 41*(4), 895–942. https://doi.org/10.1007/s00291-019-00555-y.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research, 112*(1), 3–41. https://doi.org/10.1016/S0377-2217(98)00204-5.

Burkard, R. E., & Hatzl, J. (2005). Review, extensions and computational comparison of MILP formulations for scheduling of batch processes. *Computers & Chemical Engineering, 29*(8), 1752–1769. https://doi.org/10.1016/j.compchemeng.2005.02.037.

Burkard, R. E., & Hatzl, J. (2006). A complex time based construction heuristic for batch scheduling problems in the chemical industry. *European Journal of Operational Research, 174*(2), 1162–1183. https://doi.org/10.1016/j.ejor.2005.03.011.

Carlier, J., Moukrim, A., & Sahli, A. (2018). Lower bounds for the event scheduling problem with consumption and production of resources. *Discrete Applied Mathematics, 234*, 178–194. https://doi.org/10.1016/j.dam.2016.05.021.

Carlier, J., Moukrim, A., & Xu, H. (2009). The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm. *Discrete Applied Mathematics, 157*(17), 3631–3642. https://doi.org/10.1016/j.dam.2009.02.012.

Davari, M., Ranjbar, M., de Causmaecker, P., & Leus, R. (2020). Minimizing makespan on a single machine with release dates and inventory constraints. *European Journal of Operational Research, 286*(1), 115–128. https://doi.org/10.1016/j.ejor.2020.03.029.

EnBW Kernkraft GmbH (2018). Stilllegung und Abbau von Anlagenteilen des Kernkraftwerks Philippsburg Block 2 Decommissioning and dismantling of plant components of the Philippsburg nuclear power plant, reactor no. 2: Kurzbeschreibung Brief description. https://um.baden-wuerttemberg.de/fileadmin/redaktion/m-um/intern/Dateien/Dokumente/3_Umwelt/Kernenergie/Genehmigungsverfahren/KKP/KKP_2/180116_Kurzbeschreibung_Stilllegung_und_Abbau_KKP2.pdf.

Franck, B., Neumann, K., & Schwindt, C. (2001). Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling. *OR Spectrum, 23*(3), 297–324. https://doi.org/10.1007/PL00013356.

Gerhards, P. (2020). The multi-mode resource investment problem: A benchmark library and a computational study of lower and upper bounds. *OR Spectrum, 42*(4), 901–933. https://doi.org/10.1007/s00291-020-00595-9.

Kallrath, J. (2002). Planning and scheduling in the process industry. *OR Spectrum, 24*(3), 219–250. https://doi.org/10.1007/s00291-002-0101-7.

Kelley, J. E. (1961). Critical-path planning and scheduling: Mathematical basis. *Operations Research, 9*(3), 296–320. https://doi.org/10.1287/opre.9.3.296.

Kolisch, R. (1996a). Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management, 14*(3), 179–192. https://doi.org/10.1016/0272-6963(95)00032-1.

Kolisch, R. (1996b). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research, 90*(2), 320–333. https://doi.org/10.1016/0377-2217(95)00357-6.

Kolisch, R., & Sprecher, A. (1996). PSPLIB - a project scheduling problem library. *European Journal of Operational Research, 96*, 205–216. https://doi.org/10.1016/S0377-2217(96)00170-1.

Kolisch, R., & Sprecher, A. (2021). Project scheduling problem library - PSPLIB [dataset]. Accessed January 26, 2021, http://www.om-db.wi.tum.de/psplib/main.html.

Kolisch, R., Sprecher, A., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science, 41*(10), 1693–1703. https://doi.org/10.1287/mnsc.41.10.1693.

Kondili, E., Pantelides, C. C., & Sargent, R. (1993). A general algorithm for short-term scheduling of batch operations—i. MILP formulation. *Computers & Chemical Engineering, 17*(2), 211–227. https://doi.org/10.1016/0098-1354(93)80015-F.

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2013). Comparison of mixed integer linear programming models for the resource-constrained project scheduling problem with consumption and production of resources. *Flexible Services and Manufacturing Journal, 25*(1–2), 25–47. https://doi.org/10.1007/s10696-012-9152-5.

Laborie, P. (2003). Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence, 143*, 151–188. https://doi.org/10.1016/S0004-3702(02)00362-4.

Laborie, P. (2009). IBM ILOG CP Optimizer for detailed scheduling illustrated on three problems. In W. J. van Hoeve, & J. N. Hooker (Eds.), *Integration of AI And OR techniques in constraint programming for combinatorial optimization problems* (pp. 148–162). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-01929-6_12.

Laborie, P., Rogerie, J., Shaw, P., & Vilím, P. (2018). IBM ILOG CP Optimizer for scheduling. *Constraints : An International Journal, 23*(2), 210–250. https://doi.org/10.1007/s10601-018-9281-x.

Morsy, E., & Pesch, E. (2015). Approximation algorithms for inventory constrained scheduling on a single machine. *Journal of Scheduling, 18*(6), 645–653. https://doi.org/10.1007/s10951-015-0433-1.

Neumann, K., Nübel, H., & Schwindt, C. (2000). Active and stable project scheduling. *Mathematical Methods of Operations Research, 52*(3), 441–465. https://doi.org/10.1007/s001860000092.

Neumann, K., & Schwindt, C. (1997). Activity-on-node networks with minimal and maximal time lags and their application to make-to-order production. *OR Spectrum, 19*(3), 205–217. https://doi.org/10.1007/BF01545589.

Neumann, K., & Schwindt, C. (2002). Project scheduling with inventory constraints. *Mathematical Methods of Operations Research, 56*, 513–533. https://doi.org/10.1007/s001860200251.

Neumann, K., Schwindt, C., & Trautmann, N. (2005). Scheduling of continuous and discontinuous material flows with intermediate storage restrictions. *European Journal of Operational Research, 165*(2), 495–509. https://doi.org/10.1016/j.ejor.2004.04.018.

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* (2nd). Berlin: Springer.

Schwindt, C. (1999). A branch-and-bound algorithm for the project duration problem subject to temporal and cumulative resource constraints. In *Proceedings of international conference on industrial engineering and production management*. Glasgow.

Schwindt, C., Fink, R., & Trautmann, N. (2007). A priority-rule based method for scheduling in chemical batch production. In M. Helander (Ed.), *Proceedings of the IEEE international conference on industrial engineering and engineering management* (pp. 1347–1351). IEEE Service Center. https://doi.org/10.1109/IEEM.2007.4419412.

Schwindt, C., & Trautmann, N. (2000). Batch scheduling in process industries: An application of resource-constrained project scheduling. *OR Spectrum, 22*, 501–524. https://doi.org/10.1007/s002910000042.

Sourd, F., & Rogerie, J. (2005). Continuous filling and emptying of storage systems in constraint-based scheduling. *European Journal of Operational Research, 165*(2), 510–524. https://doi.org/10.1016/j.ejor.2004.04.019.

Voß, S., & Witt, A. (2007). Hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements: A real-world application. *International Journal of Production Economics, 105*(2), 445–458. https://doi.org/10.1016/j.ijpe.2004.05.029.