Robust Optimization of Schedules Affected by Uncertain Events

Originally published in:
Journal of Optimization Theory and Applications 171(3), https://doi.org/10.1007/s10957-016-0920-3

J Optim Theory Appl (2016) 171:1033–1054
DOI 10.1007/s10957-016-0920-3

Robust Optimization of Schedules Affected by Uncertain Events

Robin Vujanic[1] · **Paul Goulart**[2] · **Manfred Morari**[1]

Received: 22 July 2015 / Accepted: 4 March 2016 / Published online: 28 March 2016

Abstract In this paper, we present a new method for finding robust solutions to mixed-integer linear programs subject to uncertain events. We present a new modeling framework for such events that result in uncertainty sets that depend parametrically on the decision taken. We also develop results that can be used to compute corresponding robust solutions. The usefulness of our proposed approach is illustrated by applying it in the context of a scheduling problem. For instance, we address uncertainty on the start times chosen for the tasks or on which unit they are to be executed. Delays and unit outages are possible causes for such events and can be very common in practice. Through our approach, we can accommodate them without altering the remainder of the schedule. We also allow for the inclusion of recourse on the continuous part of the problem, that is, we allow for the revision of some of the decisions once uncertainty is observed. This allows one to increase the performance of the robust solutions. The proposed scheme is also computationally favorable since the robust optimization problem to be solved remains a mixed-integer linear program, and the number of integer variables is not increased with respect to the nominal formulation. We finally apply the method to a concrete batch scheduling problem and discuss the effects of robustification in this case.

Communicated by Christodoulos Floudas.

✉ Robin Vujanic
vujanicr@control.ee.ethz.ch

Paul Goulart
paul.goulart@eng.ox.ac.uk

[1] Automatic Control Laboratory, ETH Zurich, Zurich, Switzerland

[2] Department of Engineering Science, University of Oxford, Oxford, England, UK

Keywords Robust optimization · Mixed-integer optimization · Scheduling · Uncertain events

Mathematics Subject Classification 49J53 · 49K99

1 Introduction

The task of robust optimization is to take the known data of an uncertain optimization problem together with information available on the uncertainty, such as bounds on the values that uncertain terms may attain, and then formulate and solve a so-called robust counterpart (RC). The RC is *another* optimization problem and, in contrast to the initial uncertain program, it is deterministic. It is constructed in such a way that its solutions remain feasible for the initial uncertain problem for any possible realization of the uncertainty.

In this paper we discuss the computation of robust solutions to uncertain linear models entailing integer variables, i.e., uncertain mixed-integer linear programs (MILPs). We propose a framework consisting of both a new concept for modeling uncertainty and the appropriate RC necessary to determine robust solutions. In contrast to the usual setting of robust optimization [1–3] in which uncertainty affects the *data* of the problem, here we consider the case in which uncertain events directly affect the *decisions taken*. To illustrate this difference and the capabilities of our results, we focus on examples involving scheduling problems. In such problems, uncertain events may for instance represent delays on the decided start time of the tasks. Delayed execution is a problem that arises frequently in practice and is an issue exacerbated by the fact that optimization techniques favor schedules that tightly pack processes in certain time windows, e.g., when prices are lowest. Representing such an event as uncertainty on the data of the optimization program is not straightforward. This difficulty is illustrated in Sect. 2 with a simple example that is later used to discuss how the same issue can be addressed using the method we propose. Beyond delays, we will discuss how our proposed architecture can be used to represent more complex uncertain outcomes, which in turn allow one to encode a diverse range of unexpected events. We discuss these possibilities on a second, more realistic scheduling model based on discrete time State-Task-Networks (STNs), which are common for scheduling short-term batch operations.

Our new modeling concept results in uncertainty sets that depend on the decision taken, also known as endogenous uncertainty [4,5]. For the computation of robust solutions we provide a system that combines a preventive as well as a reactive mechanism. Preventive action is achieved by immunizing the integer part of the problem against the decision-dependent uncertainty, while the reactive component is realized through the introduction of recourse on the continuous decisions and allows one to limit the amount of necessary conservatism. In the scheduling context, the preventive action *guarantees* that the core of the schedule remains unchanged under any contingency, that is, uncertain events can be accommodated without disrupting the remainder of the schedule. The reactive component improves the performance of the solution by adapting the batch sizes to the observed uncertainty.

1.1 Literature

The first work introducing the ideas of robust optimization appears to be [6], while the main technique of robust optimization, in which an argument based on duality is used to replace worst-case performance over an uncertain quantity with the existence of appropriate multipliers, was proposed in [7]. The robust optimization framework is most useful when the RC can be formulated as a finite optimization problem, in which is sometimes also referred to as the *explicit* RC. Explicit RCs are known for a number of practically important cases; the references [2,8–10] contain many of these results. Some of these RCs are also immediately applicable to models entailing integer variables as noted in [2, Remark 1.2., p. 26] and can be used in conjunction with our results.

Much research effort in this area has been directed toward making robust optimization approaches less conservative. One direction has been to look for new uncertainty models. The papers [1,11] address over-conservatism by considering ellipsoidal uncertainty sets. In [12], the authors investigate a problem in which the amount of uncertainty is "budgeted." Conservatism is reduced by assuming that, while all the data may be affected by uncertainty, within a certain realization only a subset of the data is actually changed from its nominal value. A second line of research has been directed toward introducing the possibility of recourse, in particular in the context of multistage decision problems; see [13–15]. An optimal recourse policy in this setting can be obtained using dynamic programming, which can become computationally intractable even for modest model sizes. This difficulty can be overcome by restricting the class of policies considered in the optimization, in particular to affine policies [16–18]. In this case, a trade-off is made between performance and computational tractability. We use these ideas in the present paper.

For the particular application of scheduling under uncertainty, approaches can be broadly classified according to whether they are *preventive* or *reactive*.

Preventive approaches based on robust optimization use and adapt the frameworks discussed so far. An application to an air separation plant is presented in [19] where the uncertainties arise due to the integration of the plant in electricity market mechanisms. The authors of [20] address uncertainties from a bounded set affecting the coefficients of the objective function and the constraints of a scheduling problem, and their scheme is verified on the same example discussed in this paper. In a follow-up paper [21], the authors also discuss the unbounded uncertainty case. In [22], the problem of quantifying the schedule resilience to uncertain demands is addressed by proposing several robustness metrics and an approach to improve schedules by considering the extreme points of the uncertainty set is proposed. In [23], a multi-objective approach is developed in which the uncertain quantities are parametrized and the Pareto-optimal surface derived can be used to assess the trade-off between multiple objectives as a function of the uncertainty realization.

The major advantage of preventive approaches is that they ensure some degree of immunity of the schedule to the uncertainties.

In reactive approaches, part of the plan may be decided in advance, but it is possible to revise the schedule in the near short term based on the observation of the unexpected events. In the approach proposed in [24], uncertainty on the execution length of tasks

is handled using a shifting algorithm that is simple to implement, but is limited in the type of uncertainties it can address. The authors of [25] use the same STN framework discussed in this paper and implement a rolling horizon procedure that is adapted to ensure fulfillment of due dates. The works [26,27] consider the possibility of changing starting times, unit reallocations and other rescheduling mechanisms to restore schedule consistency after an unexpected event. A common problem with these reactive schemes is that the new plan may be completely different from the initial one, which can be highly impractical. To alleviate this issue, [28] considers machine outages and rush order arrivals, and develops an approach in which an appropriate penalty term is introduced in the objective function that minimizes the deviation of the revised schedule from the original one. The scheme presented in [29] includes a knowledge-based expert system with the aim of minimizing the impact on the schedule from uncertainty on processing time or unit availability. For this same type of uncertainty, [30] develops a least-impact heuristic algorithm. Another heuristic is proposed in [31] in which schedule revisions are minimized by fixing binary variables according to a set of rules established to reflect production needs.

The main advantage of reactive approaches is that they allow one to achieve good performance despite the presence of uncertainties, but the required real-time flexibility may be difficult to obtain in real applications.

The scheme proposed in this paper lies between these two types of approaches. It is composed of a preventive part, which concerns the core decisions of the schedule (binary variables), and a reactive part determined by the affine recourse policy. It thereby inherits the main advantages of both approaches.

1.2 Structure of the Paper

We start in Sect. 2 with an introductory motivating example. In Sect. 3, we describe the class of uncertain optimization problems of interest and our particular uncertainty model. Since these do not fit traditional robust optimization frameworks, we derive a new RC in Sect. 3.1. In the second part of the paper we show how these results can be of practical interest in the context of scheduling under uncertainty. In Sect. 4, we explain how uncertain events affecting the decisions can be modeled within our framework and then discuss the steps necessary to compute the corresponding RC. Finally, in Sect. 5, we apply our scheme to a more realistic scheduling model, discuss the modeling of more complex uncertain outcomes and report our simulation results with particular emphasis on their interpretation.

2 A Motivating Example

Consider a problem in which the minimum cost for the execution of two tasks must be determined. Task 2 should not start before Task 1 and should also not be executed later than one time step after Task 1 is completed. The execution time for the tasks is $l_1, l_2 \in \mathbb{N}$, the scheduling horizon is $n \in \mathbb{N}$ and the cost of execution at a given time step is $c_1, c_2 \in \mathbb{R}^n$. Furthermore, executing the tasks consumes the amount $r_{j,t}$, $j \in \{1, 2\}$, $t \in \{1, \ldots, n\}$ of a resource that is available only in the limited quantity

given by R_t. We can model the problem as

$$\begin{aligned} \min_{x} \quad & (c_1^\top x_1 + c_2^\top x_2) \\ \text{subject to} \quad & \sum_{t=1}^{n} x_{j,t} = 1, \qquad j = 1, 2 \\ & \sum_{t=1}^{n} t \cdot (x_{2,t} - x_{1,t}) \geq 0 \\ & \sum_{t=1}^{n} t \cdot (x_{2,t} - x_{1,t}) \leq l_1 + 1 \\ & r_{1,t} x_{1,t} + r_{2,t} x_{2,t} \leq R_t \qquad t = 1, \ldots, n \\ & x_1, x_2 \in \{0, 1\}^n . \end{aligned} \tag{1}$$

The uncertain event we consider is a possible delay of one time step in the execution of Task 1. It is, however, unclear how this type of event could be encoded systematically within a traditional framework for robust optimization where the uncertainty is on the data alone. One may, for instance, encode the delay as uncertainty on the execution length l_1, i.e., let l_1 belong to the interval $\mathcal{L}_1 = [\hat{l}_1, \hat{l}_1 + 1]$, where $\hat{l}_1$ is the nominal value. The only part of the problem affected by this is the constraint on maximum delay. However, in this case, the worst-case scenario is when the execution time l_1 is shortest, i.e., when the realization of $l_1 \in \mathcal{L}_1$ is $\hat{l}_1$. The RC is thus identical to (1), but the solutions it returns do not support delays robustly. In particular, the sequentiality of tasks 1 and 2 could be violated. The constraint limiting the total resource consumption is not immunized against delays either. The reason why this approach does not work in this case is because encoding delays as uncertain execution times is not the right thing to do. In Sect. 4, we discuss how the problem in this example can be addressed using our proposed method.

3 Optimization Under Decision-Dependent Uncertainty

Consider the following uncertain mixed-integer linear program:

$$\text{(UP)} : \begin{cases} \min & c_x^\top x + c_y^\top y \\ \text{subject to} & Ax + By + Dw \leq b \quad w \in \mathcal{W}(x) \\ & x \in \{0, 1\}^n \end{cases}$$

where $A \in \mathbb{R}^{m \times n_x}$, $B \in \mathbb{R}^{m \times n_y}$, $b \in \mathbb{R}^m$, $c_x \in \mathbb{R}^{n_x}$, $c_y \in \mathbb{R}^{n_y}$ and $D \in \mathbb{R}^{m \times n_w}$. We assume that the uncertainty set $\mathcal{W}(x)$ in our problem is dependent on the decision variable x, i.e., $\mathcal{W}$ is a set-valued mapping $\mathcal{W} : \mathbb{R}^n \rightrightarrows \mathbb{R}^{n_w}$. This uncertainty model differs from most other literature on robust optimization [2,3] where the problem uncertainty is typically assumed to be a static set.

Uncertainty arises because the realization of the term $w \in \mathcal{W}(x)$ is unknown at the time when the problem is to be solved, although we have some influence over the set from which the uncertain terms are drawn. We aim to deal with this uncertainty in a *robust* fashion, i.e., we would like to find a solution $x^\star$ that remains feasible for the constraints in (UP) for any possible realization of $w \in \mathcal{W}(x^\star)$.

Models of uncertainty that depends on the decision taken have appeared previously in the literature, for example in the context of stochastic optimization applied to the planning of offshore oil or gas field infrastructure [5,32]. Note that the RC to this problem, in the general case where $\mathcal{W}$ is any arbitrary set-valued map, typically leads to very difficult optimization problems. In the present work, we consider uncertainty models in the particular form

$$\mathcal{W}(x) = \oplus_{k=1}^{n_x} x_k \cdot \mathcal{W}_k, \tag{2}$$

in which the $\oplus$ operator indicates the Minkowski sum of sets, each $\mathcal{W}_k$ is a (static) finite set of vectors in $\mathbb{R}^{n_w}$, and the multiplication means the product of each element in $\mathcal{W}_k$ by the scalar quantity x_k, the k-th component of the vector $x \in \mathbb{R}^{n_x}$.

We allow for the possibility of recourse on the continuous part of the problem, i.e., we can adapt the continuous decisions y depending on the observed uncertainty outcome. The particular functional dependency we choose is the affine model

$$y = Yw + v \;\; w \in \mathcal{W}(x) \tag{3}$$

where $Y \in \mathbb{R}^{n_y \times n_w}$ and $v \in \mathbb{R}^{n_y}$ are new optimization variables. This type of recourse policy is common and has already been used in other contexts; see, e.g., [2,16,18,33].

In Sect. 4, we show how the uncertainty structure (2) may be of practical interest in the context of scheduling under uncertainty. In the next section, we show that it is possible to compute a robust solution to (UP) with uncertainty structured as in (2) and under the affine policy (3) by solving a mixed-integer linear program with the same number of integer variables of (UP).

3.1 The Explicit Robust Counterpart

In the following theorem, we determine how to compute a solution to the uncertain problem (UP) that robustly satisfies the constraints and optimizes the nominal objective.

Theorem 3.1 *A robust feasible solution to* (UP) *that optimizes the nominal objective under affine recourse* (3) *can be found by solving the following deterministic, finite dimensional optimization problem*

$$\text{(RCP)}: \begin{cases} \min_{x,v,Y,\Phi,\Psi} & c_x^\top x + c_y^\top v \\ \text{subject to} & Ax + Bv + \Phi \boldsymbol{I}^{n_x} \leq b \\ & \boldsymbol{I}^{n_k \times m} \cdot \operatorname{diag}(\psi_k) \geq [(BY + D)W_k]^\top & \forall k = 1, \ldots, n_x \\ & 0 \leq \phi_k \leq x_k \overline{\psi}_k & \forall k = 1, \ldots, n_x \\ & 0 \leq \psi_k - \phi_k \leq (1 - x_k)\overline{\psi}_k & \forall k = 1, \ldots, n_x \\ & x \in \{0,1\}^{n_x}, \end{cases}$$

where $\boldsymbol{I}^{n_x} = [1, \ldots, 1]^\top \in \mathbb{R}^{n_x}$, $\boldsymbol{I}^{n_k \times m} = [\boldsymbol{I}, \ldots, \boldsymbol{I}]^\top \in \mathbb{R}^{n_k \times m}$ *and the new auxiliary optimization variables* $\Phi = [\phi_1, \ldots, \phi_k, \ldots, \phi_{n_x}] \in \mathbb{R}^{m \times n_x}$ *and*

$\Psi = [\psi_1, \ldots, \psi_k, \ldots, \psi_{n_x}] \in \mathbb{R}^{m \times n_x}$ have been introduced. $\mathcal{W}_k$ is assumed to be a finite set of vectors, n_k is the number of vectors contained in it and W_k is the matrix formed by taking these vectors as columns. The constant upper bound $\overline{\psi}_k \in \mathbb{R}^m$ has to be chosen such that

$$(BY + D)w \leq \overline{\psi}_k \ \forall w \in \mathcal{W}_k, \quad \forall k = 1, \ldots, n_x. \tag{4}$$

We call (RCP) *the* explicit robust counterpart *to* (UP).

Proof Applying our recourse model (3) to (UP) and assuming optimization toward the nominal objective results in the robust problem

$$\begin{array}{ll} \min & \left(c_x^\top x + c_y^\top (Yw + v)\right)_{w=0} \\ \text{subject to} & Ax + B(Yw + v) + Dw \leq b \ \forall w \in W(x) \\ & x \in \{0, 1\}^{n_x}. \end{array}$$

The objective simplifies to $\min_x (c_x^\top x + c_y^\top v)$. We examine the uncertain constraints row-wise, and for the i-th row obtain

$$\begin{array}{rl} & A_i x + B_i (Yw + v) + D_i w \leq b, \qquad \forall w \in W(x) \\ \iff & A_i x + B_i v + \max\limits_{w \in W(x)} \{(BY + D)_i w\} \leq b_i, \ i = 1, \ldots, m. \end{array}$$

The maximization in the expression is then treated as follows: We examine $\sup_{w \in W(x)} d_i \cdot w$ and obtain

$$\max_{w \in W(x)} \{(BY + D)_i w\} \overset{(1)}{=} \sigma_{\oplus_{k=1}^{n_x} x_k \cdot W_k} \{(BY + D)_i\} \overset{(2)}{=} \sum_{k=1}^{n_x} x_k \cdot \sigma_{W_k} \{(BY + D)_i\},$$

where

(1) follows from the definition of support function σ and the structure imposed on $W(x)$ in (2),
(2) follows from the properties of support functions that $\sigma_{X \oplus Y} = \sigma_X + \sigma_Y$; see [34, Ex. 3.35, p. 120], and that $\sigma_Y(\alpha x) = \sigma_{\alpha Y}(x) = \alpha \sigma_Y(x)$ for $\alpha \in \mathbb{R}_+$; see [35, Thm. 13.2].

We thus have

$$\max_{w \in W(x)} \{(BY + D)_i w\} = \sum_{k=1}^{n_x} x_k \cdot \max_{w \in W_k} \{(BY + D)_i w\} = \sum_{k=1}^{n_x} x_k \cdot \psi_{ik}$$

in which

$$\begin{aligned} \psi_{ik} &\geq \max_{w \in W_k} (B_i Y + D_i) w = \max_{\lambda \in \Lambda} (B_i Y + D_i) W_k \cdot \lambda \\ &= \begin{cases} \min_{\phi_i} \ \phi_i \\ \text{s.t.} \quad \mathbf{1}\phi_i \geq [(B_i Y + D_i) W_k]^\top \ \forall i = 1, \ldots, m, \ \forall k = 1, \ldots, n_x, \end{cases} \end{aligned} \tag{5}$$

where Λ is the simplex $\Lambda = \left\{\lambda \in \mathbb{R}^{n_k} \mid \mathbf{1}^\top \lambda = 1, \lambda \geq 0\right\}$ and W_k is a matrix with the vectors in $\mathcal{W}_k$ as columns. The minimization in (5) can be dropped, given its position with respect to the inequality, and the constraint can be rewritten in matrix form as

$$\mathbf{1}^{n_k \times m} \cdot \operatorname{diag}(\psi_k) \geq [(BY + D)W_k]^\top \quad \forall k = 1, \ldots, n_x.$$

Since x_k is binary, the bilinear form $x_k \psi_{ik}$ can be substituted by the continuous optimization variable z_{ik} subject to the following set of linear constraints

$$\left.\begin{array}{l} 0 \leq z_{ik} \leq x_k \overline{\psi}_{ik} \\ 0 \leq \psi_{ik} - z_{ik} \leq \overline{\psi}_{ik}(1 - x_k) \end{array}\right\} \; i = 1, \ldots, m, \;\; k = 1, \ldots, n_x,$$

where the upper bound $\overline{\psi}_{ik}$ has to be chosen according to the requirement in (4). This immediately leads to the desired robust counterpart (RCP). □

Remark 3.1 In the application examples, we will use this result, which assumes finite $\mathcal{W}_k$. It is, however, possible to handle cases in which $\mathcal{W}_k$ is a compact polyhedral. In this case, if $\mathcal{W}_k$ is expressed in vertex form

$$\mathcal{W}_k = \left\{ w \in \mathbb{R}^{n_w} : \; w = \sum_{i=1}^{n_k} w_i \lambda_i, \; \sum_{i=1}^{n_k} \lambda_i = 1, \; \lambda \geq 0 \right\},$$

then n_k is the number of vertices w_i, and W_k is the matrix formed by taking these vertices vectors as columns.

Solving the deterministic optimization problem (RCP) produces an optimal solution $(x^\star, v^\star, Y^\star, \Phi^\star, \Psi^\star)$. A robust solution to the original uncertain program (UP) is then given by $(x^\star, y^\star)$, where $y^\star$ can be determined once the uncertainty w has been observed by computing $y^\star = Y^\star w + v^\star$, according to the affine recourse policy (3). From a computational perspective, (RCP) is a mixed-integer linear program and has the same number of integer variables as the nominal version of (UP). In applications it is likely that $W_k = 0$ for many values of k, and in such cases the dimension of (RCP) can be reduced by dropping the corresponding columns in Φ and Ψ (or setting them to 0).

Remark 3.2 The objective of the optimization in Theorem 3.1 is assumed to be the nominal performance (i.e., the performance of the solution when $w = 0$). It is straightforward to optimize for the worst case instead (i.e., the performance of the solution for the worst possible $w \in \mathcal{W}(x)$), by rewriting (UP) as

$$\left\{ \begin{array}{lll} \min & t & \\ \text{subject to} & \begin{bmatrix} c_x^\top \\ A \end{bmatrix} x + \begin{bmatrix} c_y^\top & -1 \\ B & \mathbf{0}^{m \times 1} \end{bmatrix} \begin{bmatrix} y \\ t \end{bmatrix} + \begin{bmatrix} \mathbf{0}^{1 \times n_w} \\ D \end{bmatrix} w \leq \begin{bmatrix} 0 \\ b \end{bmatrix} & w \in \mathcal{W}(x) \\ & x \in \{0, 1\}^n, & \end{array} \right.$$

and applying Theorem 3.1 to this reformulation.

So far, we have worked under the assumption that recourse is available on the entire vector y. In the application example discussed in Sect. 5, the notion of time will require us to introduce additional structure on the recourse matrix Y to ensure causality of the policy.

Finally, when recourse is not possible at all, i.e., all decisions must be made before the uncertainty realization is observed, then the result in Theorem 3.1 can be simplified as follows:

Corollary 3.1 *The explicit RC to* (UP) *when no recourse is available ($Y = 0$) is the following deterministic optimization problem*

$$\begin{array}{ll} \min & c_x^\top x + c_y^\top y \\ \text{subject to} & (A+H)x + By \leq b \\ & x \in \{0,1\}^{n_x}, \end{array} \tag{6}$$

in which the i-th row, k-th column entry of the matrix $H \in \mathbb{R}^{m \times n_x}$ is

$$H_{ik} \doteq \max_{w_k \in W_k} d_i \cdot w_k, \tag{7}$$

where d_i is the i-th row of the matrix D.

This robustification is useful for certain problems in which recourse is either not available or not necessary, like in the purely binary simple scheduling problem introduced in Sect. 2 or as in the model discussed in [36]. The major advantage in this case is that the computation of a robust solution involves an optimization program that is as hard as the original nominal problem in the sense that its dimension (number of variables and constraints) is the same.

In the next section we present an application example of this proposed robust optimization framework.

4 Robust Schedule Optimization

In Sect. 2, we showed that it may be unclear how to encode even a relatively simple uncertain outcome, such as a possible execution delay, as uncertainty on the data of the problem. Using the framework proposed in this paper, in which uncertainty is not on the *data* (e.g., prices, or measured physical quantities such as positions in space, concentrations, voltages etc.) but rather on the *decisions taken*, representing this type of uncertain outcome is a straightforward task.

Example 4.1 **(modeling the uncertainty)** Consider the scheduling problem of Sect. 2. Suppose we have determined a plan in which the execution of Task 1 is started at $t = 1$, while Task 2 is started at $t = 2$, i.e., for $n = 4$, we have $x_1 = [1, 0, 0, 0]^\top$ and $x_2 = [0, 1, 0, 0]^\top$. However, at the time of planning we do not know whether we will be able to actually implement this solution because some external event may force us to delay the starting time of Task 1 by one time step. This interaction can be encoded as

$$\hat{x}_1 = x_1 + w_1 = [1, 0, 0, 0]^\top + [-1, +1, 0, 0]^\top. \tag{8}$$

Note that x_2 is not subject to any direct perturbation. While it may be affected by possible changes of x_1 through the scheduling constraints, these interactions are automatically captured by our system. Note also that *the value of the disturbance w clearly depends on the choice x*. We can represent this systematically using the model (UP) with the uncertainty structure (2) written as

$$\mathcal{W}(x) = \oplus_{j,t}\, x_{j,t} \cdot \mathcal{W}_{j,t}, \tag{9}$$

which can be parsed as follows. First, if $x_{j,t} = 0$, then the uncertainty associated with the corresponding task being started at step t is suppressed. On the other hand, when $x_{j,t} = 1$, the uncertainty set from which w is picked is $\mathcal{W}_{j,t}$. This set contains all the possible disturbances on the operation of task j if it is started at time step t. For the particular case in this example, we thus have

$$\mathcal{W}_{1,t} = \{0, \overline{w}_t\}, \quad \mathcal{W}_{2,t} = \{0\}, \quad t \in \{1, \ldots, 4\}, \tag{10}$$

with

$$\begin{aligned}
\overline{w}_1 &= [-1, 1, 0, 0, 0, 0, 0, 0]^\top \\
\overline{w}_2 &= [0, -1, 1, 0, 0, 0, 0, 0]^\top \\
\overline{w}_3 &= [0, 0, -1, 1, 0, 0, 0, 0]^\top \\
\overline{w}_4 &= [0, 0, 0, -1, 0, 0, 0, 0]^\top,
\end{aligned}$$

where $\overline{w}_4$ indicates that if task is started at $t = 4$ it can no longer be recuperated at a later time within the scheduling horizon.

On more complex scheduling models, our architecture allows one to encode a wide variety of possible uncertain outcomes. We discuss some of these possibilities in Sect. 5. Notice that all of the uncertainty sets in the previous example include the vector 0. This is necessary in order to represent a situation in which the schedule is executed as planned. We also emphasize that it is important to have an appropriate representation of the desired uncertain event. For example, a possible delay of two time steps can be represented with a construction similar to (10), namely $\mathcal{W}_{1,1} = \{0, \overline{w}_1\}$ with

$$\overline{w}_1 = [-1, 0, 1, 0, 0, 0, 0, 0]^\top.$$

On the other hand, a delay of *up to* two time steps requires $\mathcal{W}_{1,1} = \left\{0, \overline{w}_1, \overline{w}'_1\right\}$ with

$$\begin{aligned}
\overline{w}_1 &= [-1, 1, 0, 0, 0, 0, 0, 0]^\top \\
\overline{w}'_1 &= [-1, 0, 1, 0, 0, 0, 0, 0]^\top.
\end{aligned}$$

Such differences are significant and will usually lead to different robust counterparts.

According to (8), uncertain scheduling problems within our framework are generally of the form

$$\begin{array}{ll} \min & c_x^\top x + c_y^\top y \\ \text{subject to} & A(x+w) + By \leq b \;\; w \in \mathcal{W}(x) \\ & x \in \{0,1\}^{n_x}, \end{array}$$

which is a special case of (UP) in which $D = A$. Modeling the uncertainty will output n_x sets of vectors $\mathcal{W}_k \subset \mathbb{R}^{n_x}$, which fully determine the map $\mathcal{W}(x)$. Each set $\mathcal{W}_k$ contains a number of vectors n_k determined by the number of different possible outcomes considered. Thus, the memory requirement to store the uncertainty is of the order $\mathcal{O}(n_x^2 \cdot n_k)$. In typical applications, the number n_k is small. For the one-step delay example, we have for instance $n_k = 1$ or 2, as shown in (10). Furthermore, we often have $\mathcal{W}_k = \{0\}$ for many values of k.

Once the uncertainty sets $\mathcal{W}_k$ are determined, a robust solution to the corresponding uncertain scheduling problem can be obtained using the robust counterparts derived in Sect. 3.1.

Example 4.2 **(computation of the RC)** The matrices of model (1) for $n = 4$ are given by

$$\begin{array}{ll} \min_x & c_1^\top x_1 + c_2^\top x_2 \\ \text{subject to} & \left[\begin{array}{cccc:cccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \\ \hdashline 1 & 2 & 3 & 4 & -1 & -2 & -3 & -4 \\ -1 & -2 & -3 & -4 & 1 & 2 & 3 & 4 \\ \hdashline r_{11} & 0 & 0 & 0 & r_{21} & 0 & 0 & 0 \\ 0 & r_{12} & 0 & 0 & 0 & r_{22} & 0 & 0 \\ 0 & 0 & r_{13} & 0 & 0 & 0 & r_{23} & 0 \\ 0 & 0 & 0 & r_{14} & 0 & 0 & 0 & r_{24} \end{array}\right] \left[\begin{array}{c} x_1 \\ x_2 \end{array}\right] \leq \left[\begin{array}{c} 1 \\ 1 \\ -1 \\ -1 \\ \hdashline 0 \\ \hdashline l_1 + 1 \\ \hdashline R_1 \\ R_2 \\ R_3 \\ R_4 \end{array}\right] \\ & x_1, x_2 \in \{0,1\}^4. \end{array} \tag{11}$$

The sets $\mathcal{W}_k$ are given in (10). To obtain the RC, we can use the simplified result in Theorem 3.1, since the model is purely binary. While this result holds for a generic matrix D, as noted above, we have $D = A$. The entries of H can be determined using Eq. (7). For example, entry $H_{6,4}$ (sixth row, fourth column) is

$$H_{6,4} = \max_{w_4 \in \mathcal{W}_4} a_6 \cdot w_4 = \max\{0, 4\} = 4,$$

where a_6 is the sixth row of the constraint matrix in (11). After completing these calculations, one can verify that the resulting RC is given by

$$
\begin{aligned}
\min_x \quad & c_1^\top x_1 + c_2^\top x_2 \\
\text{subject to} \quad & \left[\begin{array}{cccc|cccc}
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
-1 & -1 & -1 & \boldsymbol{0} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 \\
\boldsymbol{2} & \boldsymbol{3} & \boldsymbol{4} & 4 & -1 & -2 & -3 & -4 \\
-1 & -2 & -3 & \boldsymbol{0} & 1 & 2 & 3 & 4 \\
r_{11} & 0 & 0 & 0 & r_{21} & 0 & 0 & 0 \\
\boldsymbol{r_{12}} & r_{12} & 0 & 0 & 0 & r_{22} & 0 & 0 \\
0 & \boldsymbol{r_{13}} & r_{13} & 0 & 0 & 0 & r_{23} & 0 \\
0 & 0 & \boldsymbol{r_{14}} & r_{14} & 0 & 0 & 0 & r_{24}
\end{array}\right] \left[\begin{array}{c} x_1 \\ x_2 \end{array}\right] \leq \left[\begin{array}{c} 1 \\ 1 \\ -1 \\ -1 \\ 0 \\ l_1 + 1 \\ R_1 \\ R_2 \\ R_3 \\ R_4 \end{array}\right] \\
& x_1, x_2 \in \{0,1\}^4 ,
\end{aligned}
\tag{12}
$$

in which the changes with respect to the nominal problem (11) have been highlighted with a bold italic font. We can observe how the 0 entry on the third row, fourth column correctly prevents starting Task 1 at $t = 4$, because a delay would push its execution out of the scheduling horizon. Further, the modified entries in rows 5 and 6 force Task 2 to start at least one time step after initiating Task 1, which robustly accommodates delays as desired. Finally, the last four rows show how resource consumption can be immunized against delays. For example, the eighth row reflects that an amount of resources sufficient to start Task 1 at time step $t = 2$ should be left available even if nominally we decide to start this task at $t = 1$.

Note that even if the nominal model (1) is modified, e.g., by introducing new constraints, a new RC can still be easily recomputed. The uncertainty model remains identical, and one only needs to determine the new matrix H. Hence the approach remains straightforward to apply even on more complex scheduling models.

In the next section we investigate the application of the more sophisticated robust counterpart (RCP).

5 Case Study: Robust State-Task-Networks

In this section we further explore the capabilities of our approach based on a more realistic scheduling model. We work within the framework of State-Task-Networks (STNs), a common model for batch scheduling problems which was initially proposed in [37].

We first introduce the nominal model, then discuss the type of uncertain outcomes that we want to consider and finally draw a few interpretations from the resulting robust schedules.

5.1 Nominal Model

Our concise description of the model is based on [37,38]. As represented in Fig. 1, we are given a set of available processing units $i \in \mathcal{I}$, depicted in the lower half of the figure, a set of possible tasks $j \in \mathcal{J}_i$ that can be performed on each unit, represented by

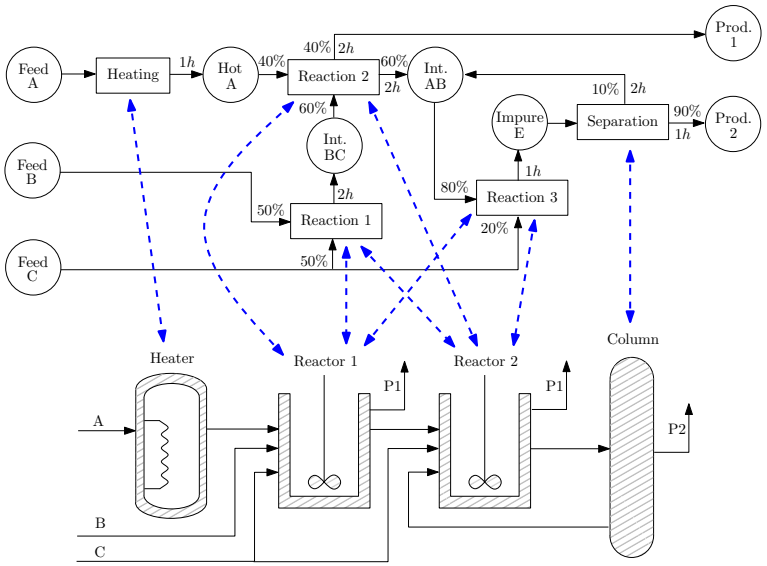

Fig. 1 Example STN structure, taken from [37,38]. The *circles* represent states (materials), while the *rectangles* are the tasks (processes). Depicted are also the units on which the tasks can be processed, and the *dashed arrows* indicate which task can be processed on which unit. Furthermore, the production recipes are shown, including material balances and processing times to obtain the output materials

the rectangles and the dashed arrows, a production recipe for each task (mass balance coefficients and processing times), and a set of given input, intermediate and output materials in a given state, indicated with the circles. We are also provided with data concerning the prices of the raw materials and the end products. To be determined are the sequence and timing of the tasks taking place in each unit within a finite time horizon divided in $t \in \mathcal{T}$ discrete steps, the amount of material undergoing each task (batch sizes), as well as the amount of raw input material purchased and final products sold.

A mathematical model for this, as given in [37], is as follows.

5.1.1 Optimization Variables

- $x_{ijt} \in \{0, 1\}$ is set to 1 if unit i starts processing task j at time step t; it is 0 otherwise.
- $y^{\text{batch}}_{ijt} \in \mathbb{R}$ is the amount of material assigned to task j in unit i at the beginning of the time step t.
- $y^{\text{state}}_{st} \in \mathbb{R}$ is the amount of material stored in state $s \in \mathcal{S}$ at the beginning of period t.

Thus, in the dimensions given for the generic problem (UP), $n_x = |\mathcal{I}| \cdot |\mathcal{J}| \cdot |\mathcal{T}|$, and $n_y = (|\mathcal{I}| \cdot |\mathcal{J}| + |\mathcal{S}|) \cdot |\mathcal{T}|$, where $\mathcal{J} = \cup_{i \in I} \mathcal{J}_i$. We have modified the notation in [37] to be consistent with our formulation (UP): x is boolean, while y is the continuous part of the problem.

5.1.2 Constraints

– Allocation constraints ensure that at a given time step, a unit can start to process at most 1 task. As long as the unit is processing this task, no other task can be started on the same unit. This can be written as

$$\begin{aligned} &\sum_{j\in\mathcal{J}_i} x_{ijt} \le 1 && \forall i,t \\ &\sum_{j\in\mathcal{J}_i}\sum_{t'=t}^{t+\tau_j-1} x_{ijt'} - 1 \le M_{ij}(1-x_{ijt}) && \forall i,t,j\in\mathcal{J}_i, \end{aligned} \tag{13}$$

where $\mathcal{J}_i$ is the set of tasks j that can be performed on unit i, $\tau_j = \max_s \tau_{js}$, and τ_{js} is the number of time steps required to produce material in state s with process task j. M_{ij} is a sufficiently large positive number. The second constraint implies that if $x_{ijt} = 1$, then for all the time steps $t' = t+1,\ldots,t+\tau_j-1$ and for all $j\in\mathcal{J}_i$, $x_{ijt} = 0$.

– Capacity limitations on the amount of material undergoing process j

$$\begin{aligned} &x_{ijt}\cdot V_{ij}^{\min} \le y_{ijt}^{\text{batch}} \le x_{ijt}\cdot V_{ij}^{\max} && \forall j,t,i\in\mathcal{I}_j \\ &0 \le y_{st}^{\text{state}} \le C_s && \forall s,t, \end{aligned} \tag{14}$$

where $\mathcal{I}_j$ is the set of units capable of performing task j, $V_{ij}^{\min}$ and $V_{ij}^{\min}$ are the capacities of processing task j on unit i, and C_s is the maximum capacity for storing state s.

– States update equation, derived from material balances

$$y_{st}^{\text{state}} = y_{s,t-1}^{\text{state}} + \sum_{j\in\bar{\mathcal{J}}_s}\bar{\rho}_{js}\sum_{i\in\mathcal{I}_j} y_{ij,t-\tau_{js}}^{\text{batch}} - \sum_{j\in\mathcal{J}_s}\rho_{js}\sum_{i\in\mathcal{I}_j} y_{ijt}^{\text{batch}} + R_{st} - D_{st} \quad \forall s,t, \tag{15}$$

where $\mathcal{J}_s$ is the set of tasks consuming material in state s, and the constant ρ_{js} is the proportion of input from state $s\in\mathcal{S}_j$ used by task j, so that $\sum_{s\in\mathcal{S}_j}\rho_{js} = 1$. In an analogous way, $\bar{\mathcal{J}}_s$ is the set of tasks producing material in state s, the constant $\bar{\rho}_{js}$ is the proportion of output in state $s\in\bar{\mathcal{S}}_j$ produced by task j. D_{st} is the demand, over time, of material in state s, and finally, R_{st} is the quantity of raw material in feed state s at time t.

5.1.3 Objective

The objective is to optimize the economic performance of the system. Profit maximization can be expressed as follows:

$$\max \sum_{s\in\mathcal{S}} \underbrace{c_{s,|\mathcal{T}|}\cdot y_{s,|\mathcal{T}|}^{\text{state}}}_{\text{value of end products}} - \underbrace{c_{s0}\cdot y_{s0}^{\text{state}}}_{\text{cost of feedstocks}}, \tag{16}$$

where c_{st} is the unit price of material in state s at time t.

Notice that the resulting STN model

$$(\mathrm{P_{STN}}): \begin{cases} \max & \text{profits (16)} \\ \text{subject to} & \text{allocation constraints (13)} \\ & \text{unit capacity limits (14)} \\ & \text{states update equations (15)} \\ & x \in \{0,1\}^{n_x} \end{cases}$$

fits the nominal version of our generic mixed-integer model (UP). The authors of [37] discuss several ways in which this nominal model could be further enriched, for instance with constraints related to usage of utilities or cleaning requirements. Furthermore, the big-M formulation in (13) is often replaced with other models that are computationally more efficient. All of these additional modeling details fit naturally into our prototype problem (UP), but are excluded for clarity of exposition.

5.1.4 Data of the Nominal Instance

We use the example batch processing scheme and data introduced in [37] to construct an instance of the nominal problem. We summarize it here for completeness. The available processing units are a heater, two reactors 1 and 2, and a column, as shown in Fig. 1. The heater can only perform the heating task, the column only separation, while reactors 1 and 2 can perform any of the three reactions. The plant is used to process the input materials A, B and C into the final products 1 and 2. The material balances for each task (ρ_{js}, $\bar{\rho}_{js}$), as well as the processing times τ_{js} are reported on the figure, while the processing and storage capacities $V_{ij}^{\max}$ and C_s are reported in Table 1, and $V_{ij}^{\min} = 0$ kg, $\forall i, j$. The time horizon considered is of 10 h divided in steps of 1 h each. For the objective function, we assign a value of 10 units/kg for each of the two final products 1 and 2, intermediates are assigned a value of -1, since it is undesirable to have them left in storage at the end of the horizon, and raw materials have a cost of 0.

Table 1 Parameter values for the nominal problem instance

Parameter value (kg)				
$C_{\mathrm{A,B,C,prod1,prod2}}$	C_{hotA}	C_{intAB}	C_{intBC}	C_{impureE}
$+\infty$	100	200	150	100
$V_{\mathrm{heater},j}^{\max}$	$V_{\mathrm{reactor1},j}^{\max}$	$V_{\mathrm{reactor2},j}^{\max}$	$V_{\mathrm{column},j}^{\max}$	
100	80	50	20	$\forall j \in \mathcal{J}$

5.2 Uncertain Outcomes Considered and Schedule Robustification

Using our system, we can represent a diverse range of uncertain outcomes affecting the nominal schedule $\mathrm{P_{STN}}$. With arrangements similar to (8) it is possible to encode, for instance, uncertainty on which unit a task will be processed, or that instead of processing a given task on a given unit, a sequence of other tasks on other units may be started instead. It is also possible to limit these uncertain outcomes to particular time windows. These options allow one to model complex uncertainty outcomes. As a concrete example for the chemical plant in Fig. 1, we may encode the fact that instead of running reaction 1 on reactor 1, we may want to run reaction 2 on the same reactor starting on the same time step, followed one time step later by reaction 1 on reactor 2, and that this swap may happen only in the first half of the scheduling horizon. Specific time windows in this case may help encode the fact that, for instance, task rescheduling can only occur during certain hours of the day. On longer planning horizons, one can also encode alternatives in which certain processes are run overnight instead of during the day. All of this can be represented by an appropriately constructed perturbation of x_{ijt}, as in (8).

In the following, we present the results of our numerical experiments under two possible scenarios. In the first scenario, the execution of heating tasks by the heater can be delayed by one time step (1 h) any time. In the second scenario, we allow for swapping the execution of reaction 2 from reactor 1 to reactor 2 any time after the first 4 h.

To achieve this, we let w act on the decisions according to the following uncertain version of $\mathrm{P_{STN}}$

$$\begin{aligned}
&\sum_{j\in\mathcal{J}_i}(x_{ijt}+w_{ijt}) \le 1 && \forall i,t\\
&\sum_{j\in\mathcal{J}_i}\sum_{t'=t}^{t+\tau_j-1}(x_{ijt'}+w_{ijt'})-1 \le M_{ij}(1-(x_{ijt}+w_{ijt})) && \forall i,t,\ j\in\mathcal{J}_i\\
&(x_{ijt}+w_{ijt})\cdot V_{ij}^{\min} \le y_{ijt}^{\mathrm{batch}} \le (x_{ijt}+w_{ijt})\cdot V_{ij}^{\max} && \forall j,t,\ i\in\mathcal{I}_j,
\end{aligned} \tag{17}$$

and where all of the other equations are identical to the nominal model, including the objective function (we opt to optimize for the nominal objective). The construction of uncertainties w that model possible execution delays is essentially the same as in (10). This shows that modeling uncertain events within our framework remains straightforward even on more complex scheduling systems. For unit swaps, the construction is analogous. The uncertain equations remain the same as in (17), and $\mathcal{W}_{ijt}=\{0,\overline{w}_{ijt}\}$ where

$$\left.\begin{aligned}\overline{w}_{ijt}[\mathrm{unit}_1,j',t']=-1,\\ \overline{w}_{ijt}[\mathrm{unit}_2,j',t']=+1,\end{aligned}\right\}\quad \forall j',\ t'\ge 4,$$

in which $\overline{w}_{ijt}[i',j',t']$ is the (i',j',t')-entry of the vector $\overline{w}_{ijt}$. All other entries of $\overline{w}_{ijt}$ are 0.

Once the appropriate uncertainty sets for these two scenarios are constructed, robust schedules can be computed using Theorem 3.1. One last arrangement concerning the

structure of the matrix Y introduced in (3) has to be made. This is to enforce causality of the policy, meaning that recourse decisions cannot depend on uncertainty realizations happening in the future. If we index the problem's variables by time, we have that

$$y_t = \sum_{t'=0}^{t-1} Y_{t,t'} \cdot w_{t'} + v_t,\ t = 0, \ldots, |\mathcal{T}|, \tag{18}$$

where $y_t = [y_{ijt'}^{\text{batch}}, y_{st'}^{\text{state}}]_{t'=t}$, $v_t \in \mathbb{R}^{n_v}$ where $n_v = |\mathcal{I}| \cdot |\mathcal{J}| + |\mathcal{S}|$ and $Y_{t,t'} \in \mathbb{R}^{n_v \times n_Y}$ with $n_Y = |\mathcal{I}| \cdot |\mathcal{J}|$. Accordingly, we must impose the following lower-triangular structure on Y:

$$y = \begin{pmatrix} 0 & & \cdots & 0 \\ Y_{1,0} & 0 & & 0 \\ \vdots & \ddots & \ddots & \vdots \\ Y_{|\mathcal{T}|-1,0} & \cdots & Y_{|\mathcal{T}|-1,|\mathcal{T}|-1} & 0 \end{pmatrix} w + v. \tag{19}$$

Note also that deploying the reactive part of the schedule in real time does not involve any difficult computations. In particular, it is not necessary to solve a new optimization problem. Namely, the problem (RCP) is solved before the schedule is started, providing the robust $x^\star$ and $v^\star$, as well as the recourse matrix $Y^\star$. During operations, as the uncertainties w_t are observed through time, one only needs to evaluate the expression (18) to determine how the continuous quantities have to be regulated to respond to the uncertain events.

5.3 Results

Computations are carried on a PC with a 2.5-GHz CPU and 24 GB RAM; the optimization problems are modeled using cvxpy [39] and solved with Gurobi 6.0.4 [40]. In both scenarios, the nominal model is solved in approximately 0.2 s, while the RC takes 1.1 s. The nominal schedule is depicted in Fig. 2 and reproduces the result in [37]. In the following, we discuss more in detail the results concerning the robust versions.

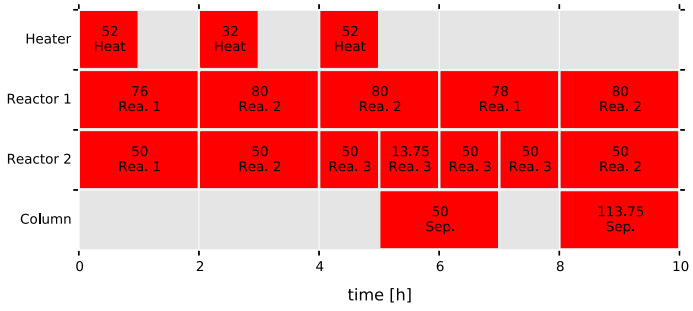

Fig. 2 Nominal batch production schedule for the example process plant shown in Fig. 1

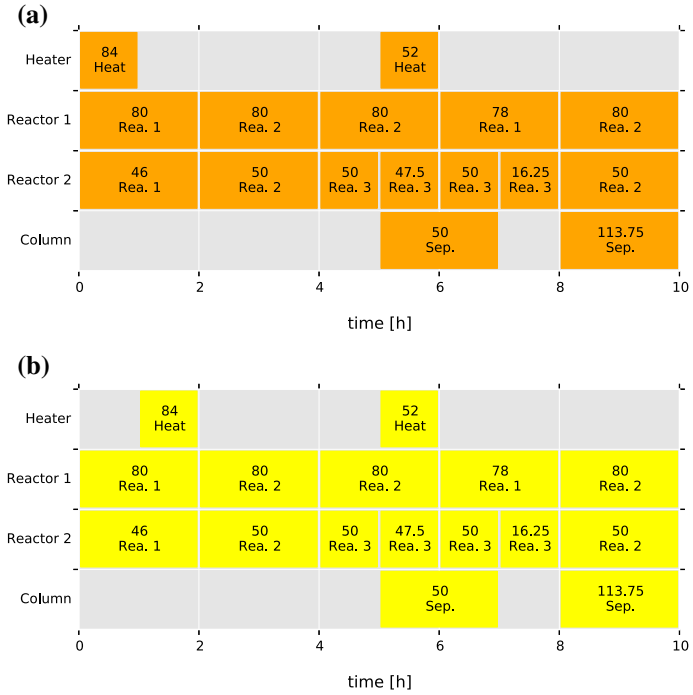

Fig. 3 Illustration of a robust schedule that can support time delays in the heating operation. **a** Robust schedule. **b** Adaptation when the external event (heating delay) occurs

5.3.1 Heater Delay

Figure 3a depicts the robust schedule under the possibility of heating delay. If we compare it with the nominal one in Fig. 2, it is evident that they only differ in the way the heater is operated.

Note that heating is used directly only by reaction 2 (see Fig. 1) and, through the intermediate products, it is also necessary for reaction 3 and the separation. Since these are executed for the first time during $t = 3$, the robust solution prescribes to preload the heater at $t = 1$ with a sufficient quantity to support these first reactions, leave at least one time step available for a possible 1 h delay by processing the remaining quantity of A into hot A at $t = 5$.

Figure 3b shows what happens to the robust schedule when an external event occurs. In this case, the heating planned for $t = 1$ is shifted to $t = 2$. This change can be accommodated because the heater is not occupied at this time, and all of the rest of the schedule is compatible with this modification. This is the most important feature of the proposed approach: The robust solution for x_{ijt} remains feasible for any disturbance realization. The decision pertinent to batch sizes (y_{ijt}^{batch}) can be made a function of the observed uncertainty realization through the affine policy (3); however, in this case, all batch sizes remain unchanged, indicating that the possibility of recourse is not exploited.

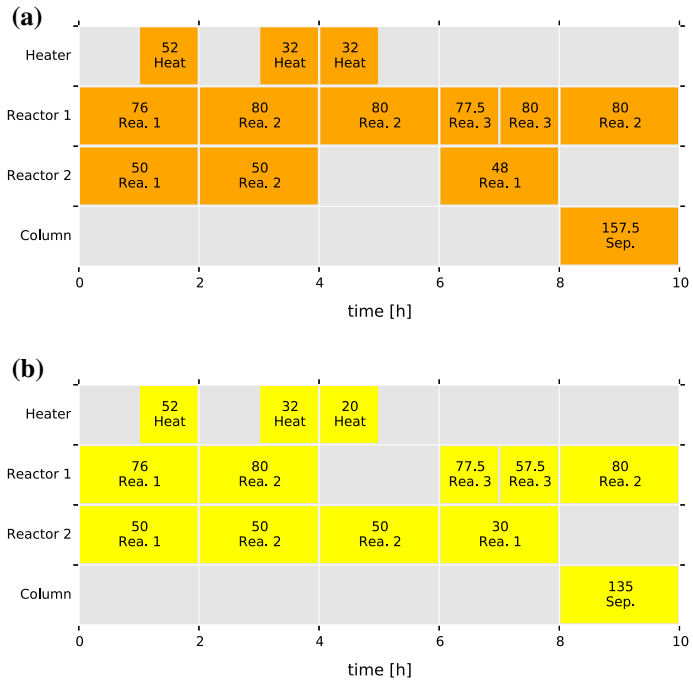

Fig. 4 Illustration of a robust schedule that can support unit exchanges: after $t = 4$ h, instead of running reaction 2 on reactor 1, it should be possible to run the same reaction on reactor 2. **a** Robust schedule. **b** Adaptation when the external event (unit swap) occurs

The nominal and the robust schedules are similar because the utilization factor of the heater is relatively low. This is also reflected in the objective function: Resorting to a robust schedule does not decrease the attained objective (2744.4 units for both nominal and robust cases).

5.3.2 Exchange of Processing Unit

In the second scenario we study the effect of exchanging processing unit for reaction 2, from reactor 1 to reactor 2, but only after the first 4 h. The results are illustrated in Fig. 4.

In this case, the robust schedule is substantially different from the nominal one, in particular after $t = 4$; see Fig. 4a. Notice that, after this time, when reaction 2 is executed on reactor 1, reactor 2 is kept free to allow for the possible exchange. Since the capacity of the second reactor is smaller, the exchange has impact on the quantities and timing processed by the interdependent tasks. This also means that intermediate products may have to be stored for longer periods, and the robust schedule ensures that storage capacities are not surpassed under any contingency.

Figure 4b confirms that the decision on x_{ijt} is unaffected by and remains feasible for the disturbance realization shifting the execution of reaction 2 from reactor 1 to reactor 2 at $t = 4$. In contrast to the previous case, however, the possibility of recourse

is exploited: since the alternative unit is smaller, the batch sizes following the uncertain outcome are reduced.

In terms of objective, since reaction 2 is necessary for producing all the final products, and the processing units (reactor 1 and 2) are heavily utilized, the introduction of the margins necessary for exchanges results in a decreased objective. In this example, the nominal objective drops from 2774.4 to 2513.8 units.

6 Conclusions

We presented a new framework for handling uncertain mixed-integer linear models with a robust approach. We proposed a new modeling system for uncertainties that are on the decision taken rather than on the data, together with the results necessary for computing the corresponding robust solutions. We then illustrated how the framework can be applied in the context of a scheduling problem subject to operational uncertainties, such as unit delays or required unit swaps.

Further lines of inquiry include the investigation of the application of the proposed framework to other scheduling systems, in particular to continuous time formulations. These could be approached by combining the more traditional results of robust optimization with the methods discussed in this paper. In Theorem 3.1, we provide a RC for generic MILPs. Another line of investigation is thus its application to problems beyond scheduling, in which the interaction expressed in (8) could be relevant.

Acknowledgements The authors are thankful to Peyman Mohajerin Esfahani from EPFL, who helped integrating recourse in our approach, as well as to Qi Zhang from Carnegie Mellon University, for his early feedback on our manuscript. This research was supported by the Swiss National Science Foundation grant P2EZP2 159089.

References

1. Ben-Tal, A., Nemirovski, A.: Robust convex optimization. Math. Oper. Res. **23**(4), 769–805 (1998)
2. Ben-Tal, A., Ghaoui, L.E., Nemirovski, A.: Robust Optimization. Princeton Series in Applied Mathematics. Princeton University Press, Princeton (2009)
3. Bertsimas, D., Brown, D.B., Caramanis, C.: Theory and applications of robust optimization. SIAM Rev. **53**, 464 (2011)
4. Pflug, G.C.: On-line optimization of simulated Markovian processes. Math. Oper. Res. **15**(3), 381–395 (1990)
5. Goel, V., Grossmann, I.E.: A class of stochastic programs with decision dependent uncertainty. Math. Program. **108**(2–3), 355–394 (2006)
6. Soyster, A.L.: Technical note—convex programming with set-inclusive constraints and applications to inexact linear programming. Oper. Res. **21**(5), 1154–1157 (1973)
7. Thuente, D.J.: Technical note—duality theory for generalized linear programs with computational methods. Oper. Res. **28**(4), 1005–1011 (1980)
8. Li, Z., Ding, R., Floudas, C.A.: A comparative theoretical and computational study on robust counterpart optimization: I. robust linear optimization and robust mixed integer linear optimization. Ind. Eng. Chem. Res. **50**(18), 10567–10603 (2011)
9. Li, Z., Tang, Q., Floudas, C.A.: A comparative theoretical and computational study on robust counterpart optimization: II. probabilistic guarantees on constraint satisfaction. Ind. Eng. Chem. Res. **51**(19), 6769–6788 (2012)

10. Li, Z., Floudas, C.A.: A comparative theoretical and computational study on robust counterpart optimization: III. improving the quality of robust solutions. Ind. Eng. Chem. Res. **53**(33), 13112–13124 (2014)
11. El Ghaoui, L., Lebret, H.: Robust solutions to least-squares problems with uncertain data. SIAM J. Matrix Anal. Appl. **18**(4), 1035–1064 (1997)
12. Bertsimas, D., Sim, M.: The price of robustness. Oper. Res. **52**(1), 35–53 (2004)
13. Kuhn, D., Wiesemann, W., Georghiou, A.: Primal and dual linear decision rules in stochastic and robust optimization. Math. Program. **130**(1), 177–209 (2011)
14. Bertsimas, D., Georghiou, A.: Design of near optimal decision rules in multistage adaptive mixed-integer optimization. Oper. Res. **63**(3), 610–627 (2015)
15. Georghiou, A., Wiesemann, W., Kuhn, D.: Generalized decision rule approximations for stochastic programming via liftings. Math. Program. **152**, 1–38 (2010)
16. Ben-Tal, A., Goryashko, A., Guslitzer, E., Nemirovski, A.: Adjustable robust solutions of uncertain linear programs. Math. Program. **99**(2), 351–376 (2004)
17. Guslitser, E.: Uncertainty-immunized solutions in linear programming. Master's thesis, Minerva Optimization Center, Technion (2002)
18. Goulart, P.J., Kerrigan, E.C., Maciejowski, J.M.: Optimization over state feedback policies for robust control with constraints. Automatica **42**(4), 523–533 (2006)
19. Zhang, Q., Grossmann, I.E., Heuberger, C.F., Sundaramoorthy, A., Pinto, J.M.: Air separation with cryogenic energy storage: optimal scheduling considering electric energy and reserve markets. AIChE J. **61**(5), 1547–1558 (2015)
20. Lin, X., Janak, S.L., Floudas, C.A.: A new robust optimization approach for scheduling under uncertainty: I. bounded uncertainty. Comput. Chem. Eng. **28**(6), 1069–1085 (2004)
21. Janak, S.L., Lin, X., Floudas, C.A.: A new robust optimization approach for scheduling under uncertainty: II. uncertainty with known probability distribution. Comput. Chem. Eng. **31**(3), 171–195 (2007)
22. Vin, J.P., Ierapetritou, M.G.: Robust short-term scheduling of multiproduct batch plants under demand uncertainty. Ind. Eng. Chem. Res. **40**(21), 4543–4554 (2001)
23. Jia, Z., Ierapetritou, M.G.: Generate pareto optimal solutions of scheduling problems using normal boundary intersection technique. Comput. Chem. Eng. **31**(4), 268–280 (2007)
24. Cott, B., Macchietto, S.: Minimizing the effects of batch process variability using online schedule modification. Comput. Chem. Eng. **13**(1), 105–113 (1989)
25. Rodrigues, M., Gimeno, L., Passos, C., Campos, M.: Reactive scheduling approach for multipurpose chemical batch plants. Comput. Chem. Eng. **20**, 1215–1220 (1996)
26. Méndez, C.A., Cerdá, J.: Dynamic scheduling in multiproduct batch plants. Comput. Chem. Eng. **27**(8), 1247–1259 (2003)
27. Mendez, C.A., Cerdá, J.: An MILP framework for batch reactive scheduling with limited discrete resources. Comput. Chem. Eng. **28**(6), 1059–1068 (2004)
28. Vin, J.P., Ierapetritou, M.G.: A new approach for efficient rescheduling of multiproduct batch plants. Ind. Eng. Chem. Res. **39**(11), 4228–4238 (2000)
29. Ruiz, D., Cantón, J., Nougués, J.M., Espuña, A., Puigjaner, L.: On-line fault diagnosis system support for reactive scheduling in multipurpose batch chemical plants. Comput. Chem. Eng. **25**(4), 829–837 (2001)
30. Kanakamedala, K.B., Reklaitis, G.V., Venkatasubramanian, V.: Reactive schedule modification in multipurpose batch chemical plants. Ind. Eng. Chem. Res. **33**(1), 77–90 (1994)
31. Janak, S.L., Floudas, C.A., Kallrath, J., Vormbrock, N.: Production scheduling of a large-scale industrial batch plant. II. Reactive scheduling. Ind. Eng. Chem. Res. **45**(25), 8253–8269 (2006)
32. Tarhan, B., Grossmann, I.E., Goel, V.: Stochastic programming approach for the planning of offshore oil or gas field infrastructure under decision-dependent uncertainty. Ind. Eng. Chem. Res. **48**(6), 3078–3097 (2009)
33. Vujanic, R., Schmitt, M., Warrington, J., Morari, M.: Extending affine control policies to hybrid systems: robust control of a DC–DC buck converter. In: European Control Conference, pp. 1523–1528 (2013)
34. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
35. Rockafellar, R.T.: Convex analysis. In: Princeton Landmarks in Mathematics. Princeton University Press, Princeton (1997). Reprint of the 1970 original, Princeton Paperbacks
36. Vujanic, R., Mariéthoz, S., Goulart, P., Morari, M.: Robust integer optimization and scheduling problems for large electricity consumers. In: American Control Conference, pp. 3108–3113 (2012)

37. Kondili, E., Pantelides, C., Sargent, R.: A general algorithm for short-term scheduling of batch operations—I. MILP formulation. Comput. Chem. Eng. **17**(2), 211–227 (1993)
38. Biegler, L.T., Grossmann, I.E., Westerberg, A.W., Kravanja, Z.: Systematic Methods of Chemical Process Design, vol. 796. Prentice Hall PTR, Upper Saddle River (1997)
39. Diamond, S., Chu, E., Boyd, S.: CVXPY: A Python-embedded modeling language for convex optimization, version 0.2. http://cvxpy.org/ (2014)
40. Gurobi: Constrained optimization software. http://www.gurobi.com/ (2014)