Article

A General State-Space Formulation for Online Scheduling

Dhruv Gupta and Christos T. Maravelias*

Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, WI, USA 53706; dgupta6@wisc.edu

* Correspondence: christos.maravelias@wisc.edu; Tel.: +1-608-265-9026

Abstract: We present a generalized state-space model formulation particularly motivated by an online scheduling perspective. Through these proposed generalizations, we enable a natural way to handle routinely encountered disturbances and a rich set of corresponding counter-decisions. Thereby, greatly simplifying and extending the possible application of mathematical programming based online scheduling solutions to diverse application settings.

Keywords: state-space model; uncertainty; mixed-integer linear programming; model predictive control; bio-manufacturing

---

1. Introduction

Scheduling plays an important role in all industrial production facilities [1]. Contingent on the scale of operation, optimization based scheduling methods can even achieve multi-million dollars increase in profits [2]. Thus, considerable effort has been devoted towards developing optimization models that accurately represent the decision making *flexibility* in these facilities [3]. Maravelias (2012) [4] provides a unified notation and a systematic framework for the description of chemical scheduling problems. Further, significant advances in solution methods, now enable us to solve small size scheduling problems. For example, a highly constrained scheduling instance over a network of 8 processing units, 19 tasks, and 26 materials, with a realistic scheduling horizon of 2 weeks, was shown to be solved to optimality in less than 1 minute on an ordinary office computer [5]. Thus, being able to generate and revise schedules in an online fashion, so as to account for new information and disturbances, is very much a reality now. The Dow Chemical Company has already adopted online scheduling in many of its production facilities [6–8].

Scheduling models, as have been developed till now, were not necessarily designed with an emphasis to be natively ready for implementation in an online scheduling setting. Thus, the online framework utilizing a model had to be tailored to that specific model, and required many ad-hoc (heuristic) adjustments to be able to represent and resolve a disturbance to the schedule [9–23]. The introduction of the state-space *idea* to chemical production scheduling alleviated many of these issues that arose from having to make ad-hoc adjustments [24].

In this work, we present a generalized and extended state-space model which is suitable for implementation in an online scheduling setting. Although, here, we focus on expanding the modeling scope, it is important to point out that once a model has been adopted, there are still many other factors which influence the performance of an online scheduling method [25]. These factors are: the online optimization horizon length, the re-computation trigger and its frequency if periodic, allowable changes from one online iteration to the next, any added constraints (e.g. terminal constraints), and the modeling of uncertainty (deterministic vs. stochastic optimization) [8].

This paper is structured as follows. In Section 2, we present a brief background on chemical production scheduling and discuss the state-space model of Subramanian et. al [24]. In Section 3, we present a reformulated state-space model, based on a new convention, and showcase the generalizations on it one at a time. In Section 4, we present the final integrated model, with all generalizations present simultaneously, which requires more than simply concatenating all the individual generalizations together. Finally, in Section 5, we demonstrate the applicability of our proposed new model to a case study taken from the field of bio-manufacturing. Throughout the text, we use lower case Latin characters for indices, uppercase Latin bold letters for sets, uppercase Latin characters for variables, and Greek letters for parameters.

2. Background

In this section, we present the necessary background to be able to follow through the new general state-space model that we propose in this paper. Here, first, we layout the general problem statement, a standard problem representation framework, and briefly describe model classification, and solution methods. Second, we show a MILP based widely adopted scheduling model. Third, we describe the typical state-space formulation adopted in MPC technology. Finally, we provide a short overview of the state-space based scheduling model pioneered by Subramanian and co-workers [24,26,27].

2.1. Chemical production scheduling

2.1.1. General problem statement

The general scheduling problem can be stated as follows. Given:

(i) Production facility data (e.g., unit capacities and connectivity),
(ii) Production recipes (e.g., processing times and mixing rules),
(iii) Production costs (e.g., material holding costs),
(iv) Material availability (e.g., raw materials delivery amounts and dates),
(v) Resource availability (e.g., maintenance schedule and utility levels), and
(vi) Production targets or orders with due-times;

scheduling seeks to find:

(i) Number and the associated processing-sizes of the needed tasks,
(ii) Assignment of these tasks to processing units, and
(iii) Timing (or just the sequence) of these tasks on the assigned units;

so as to meet production targets at minimum cost, or to maximize profit if production beyond the given target is allowed. Apart from minimization of cost, or maximization of profit, the objective can also be minimization of makespan, or minimization of earliness, or any other suitable objective for the considered application. In general, several processing characteristics and constraints could also be present such as sequence dependent changeovers, setup times, storage constraints, time-varying utility costs, etc. [1,3].

2.1.2. Problem representation

Before a scheduling problem can be solved, we need an abstract framework to represent the different elements of the problem, viz., the production facility, the associated production recipe, etc. The state task network (STN) enables this representation [28]. Under this representation, tasks are carried out on units (equipment), and they transform materials (states) from one to another. Apart from the material to be processed and the equipment to process these materials on, these tasks can also require resources, such as, utilities, manpower etc. Another popular framework, is the resource task network (RTN) [29]. In contrast to STN, in which materials, units, and utilities are treated as different from one another, in RTN, these are treated at par, all termed together as resources. We use the STN representation in this paper, but the general modeling ideas presented are also easily adaptable to the RTN representation.

The STN representation primarily comprises of tasks $i \in \mathbf{I}$, units $j \in \mathbf{J}$, and materials $k \in \mathbf{K}$. The set of tasks producing/consuming material k are denoted by $\mathbf{I}_k^+/\mathbf{I}_k^-$; task i consumes/produces material k equivalent to $\rho_{ik}/\bar{\rho}_{ik}$ mass fraction of its batch-size ($\rho_{ik} < 0$ for consumption and $\bar{\rho}_{ik} > 0$ for production). The subset of tasks that can be carried out on unit j are denoted by $\mathbf{I}_j$; The processing time of task i, when executed on unit j, is denoted by τ_{ij}. On any given unit, only one task can be performed at a time with its batch-size between lower (β_{ij}^{min}) and upper capacities (β_{ij}^{max}); the associated fixed and proportional production costs of carrying out task i on unit j are α_{ij}^{F} and α_{ij}^{P} respectively. Feed, intermediate, and product materials are denoted by $k \in \mathbf{K^F}/\mathbf{K^I}/\mathbf{K^P}$; there are possible incoming deliveries (ζ_{kt}) and outgoing orders (ξ_{kt}) at certain times for select materials; the selling price, inventory cost, and backlog cost of material k are γ_k, γ_k^{INV}, and γ_k^{BO}, respectively.

In Fig. 1, we see a process network's STN representation comprising of 4 material nodes (circular) labeled M0-M3 and 4 task nodes (rectangular) labeled T1-T4. Arcs connect task nodes with corresponding input/output material nodes. Tasks can be carried out in compatible units and could require utilities. Task-unit mapping and task batch-size capacities (β^{min}/β^{max}) are also shown here. Material prices (γ) are shown adjacent to the material nodes.

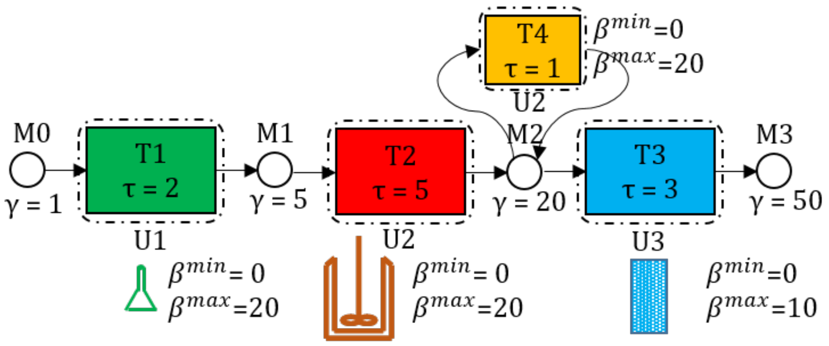


Figure 1. STN representation of a process network. This network, for a bio-manufacturing process, is described in detail in Section 5. It has three steps (tasks) for production of a pharmaceutical ingredient (material M3). T1 is the task of preparing the cell cultures in lab-size beakers. T2 denotes the task of having the cell culture grow and produce the pharmaceutical active ingredient, on a feed of sugars, in a bio-reactor. T3 is a purification task, which is carried out in chromatograph columns. Finally, T4 is a dummy task to model storage of material M2 inside unit U2. $\rho_{ik}/\bar{\rho}_{ik}$ values (not shown in the figure) are either -1 or +1 depending on whether the task is consuming the material or producing it.

2.1.3. Model classification

Scheduling models can be classified on the basis of (i) optimization decisions, (ii) modeling elements, and (iii) modeling of time [4]. Models that employ a time-grid are either continuous time or discrete time models. In discrete time models, the fixed time-grid spacing is denoted by δ. Events can take place only at these grid time-points. Thus, all time-related parameters are rounded in a conservative direction, such that the resulting schedule computed using these new parameter values is feasible even for the original parameter values. Hence, processing times and raw material delivery dates are rounded up, while due dates are rounded down so as to match with an integer multiple of δ.

Even though, having a discrete time-grid introduces the above approximation error, discrete time models have several advantages over continuous time-grid models. For example, accounting for utility consumption, inventory and backlog costs, time varying prices, or time-dependent resource availability introduces non-linearities in continuous time models, but not so in discrete time models [30]. Furthermore, discrete time models are, in general, at least as effective as continuous time models, and in fact are better suited for large scale problems with several additional processing features [31]. In this work, we employ a discrete time-grid for our state-space model.

2.1.4. Solution methods

To tackle the computational challenge of MILP scheduling models, several solution methods have been proposed: (1) tightening methods based on preprocessing algorithms and valid inequalities [32–38], (2) reformulations [5,37,39–41], (3) decomposition methods [42–47], (4) heuristics [19,48–50], and (5) hybrid methods [51–55]. Finally, parallel computing has been utilized to obtain faster solutions [56–58].

2.2. Scheduling MILP model

The discrete time STN MILP scheduling model modified from Shah et al. (1999) [59] comprises of Eqs. 1-6. Time is represented by index $t \in \mathbf{T}$. Binary variable W_{ijt}, when 1, implies task i is starting on unit j at time t. Variable $B_{ijt} \in [\beta_{ij}^{min}, \beta_{ij}^{max}]$ denotes its batch-size. The assignment constraint (Eq. 1) ensures only one task can be executed on a unit at a time.

$$\sum_{i \in \mathbf{I}_j} W_{ijt} + \sum_{i \in \mathbf{I}_j} \sum_{n=1}^{\tau_{ij}-1} W_{ij(t-n)} \leq 1 \quad \forall\, j, t \tag{1}$$

Equation 2 ensures that the batch-size of a task, if initiated, is within its upper and lower bounds.

$$\beta_{ij}^{min} W_{ijt} \leq B_{ijt} \leq \beta_{ij}^{max} W_{ijt} \quad \forall\, j, i \in \mathbf{I}_j, t \tag{2}$$

S_{kt}, which is the variable denoting inventory of material k during time-period $(t-1, t]$, is calculated in Eq. 3 as a balance of production/consumption and outgoing (V_{kt})/incoming (ζ_{kt}) shipments.

$$S_{k(t+1)} = S_{kt} + \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^+} \bar{\rho}_{ik} B_{ij(t-\tau_{ij})} + \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^-} \rho_{ik} B_{ijt} - V_{kt} + \zeta_{kt} \qquad \forall\, k, t \tag{3}$$

Equation 4 couples the outgoing shipment variable V_{kt} with demand, ξ_{kt}, for material k at time t. Backlog variables, BO_{kt}, denote pending demand during time-period $(t-1, t]$, and are penalized in the cost minimization objective function (Eq. 5).

$$BO_{k(t+1)} = BO_{kt} - V_{kt} + \xi_{kt} \quad \forall\, k, t \tag{4}$$

$$z_{\text{cost}} = \min \; \sum_k \sum_t (\gamma_k^{INV} S_{kt} + \gamma_k^{BO} BO_{kt}) + \sum_j \sum_{i \in \mathbf{I}_j} \sum_t (\alpha_{ij}^F W_{ijt} + \alpha_{ij}^P B_{ijt}) \tag{5}$$

Finally, the domain of all the variables is restricted via Eq. 6:

$$W_{ijt} \in \{0, 1\}; \; B_{ijt}, V_{kt}, S_{kt}, BO_{kt} \geq 0 \tag{6}$$

2.3. Standard form of state-space models

State-space model formulations have been useful, alongside frequency domain models, in process control [60–64]. Now, as optimization based control and economic model predictive control are becoming the new standard, state-space models have become ubiquitous [65,66]. In the most general form, a state-space based model can be written as $\frac{dx}{dt} = f(x, u, d)$; where x are the states, u are the manipulated inputs, and d are the disturbances. The function $f(\cdot)$ is not theoretically restricted to the class of linear functions, but is typically approximated as linear due to computational tractability considerations. The linear difference equation form for $f(\cdot)$ yields the model as:

$$x(t+1) = Ax(t) + Bu(t) + B_d d(t) \tag{7}$$

where, A, B, and B_d are state-space matrices and t is the index for time. The states x need not be associated with a physically identifiable entity in the plant. Some can have a direct physical meaning, while others can be artificial (e.g. augmented) constructs so as to enable the modeling exercise. The output (measurements y) is related to the states and inputs as $y = h(x, u)$, where $h(.)$ can be non-linear, but is typically linear (e.g. $y(t) = Cx(t) + Du(t)$, where C and D are coefficient matrices). The control optimization model has to follow the plant physical constraints and any other imposed constraints due to operational strategy (e.g. for environmental concerns) or those that enable better closed-loop properties (e.g. economics and stability). These constraints, when linear, can take the general form:

$$E_x x(t) + E_u u(t) + E_d d(t) \leq 0 \tag{8}$$

where, E_x, E_u, and E_d are the coefficient matrices of the states, inputs, and disturbances, respectively. If there are any equality constraints, these can also be represented as two opposite inequality constraints, so as to conform to the general form (Eq. 8). For example, the following constraints are equivalent:

$$(E_x x(t) + E_u u(t) + E_d d(t) = 0) \Leftrightarrow \left(\begin{bmatrix} E_x \\ -E_x \end{bmatrix} x(t) + \begin{bmatrix} E_u \\ -E_u \end{bmatrix} u(t) + \begin{bmatrix} E_d \\ -E_d \end{bmatrix} d(t) \leq 0 \right) \tag{9}$$

Thus, any equality constraints that we propose from here on, can be easily converted to the general inequality form through the use of the above trick. Finally, the objective function takes the form:

$$z_{cost} = \min_{u(0),u(1),...,u(N-1)} V_N(x(0), u(0), x(1), u(1), ..., x(N-1), u(N-1)) \tag{10}$$

where N is the number of discrete time-points in the online optimization horizon.

A wealth of literature focuses on the closed-loop properties of the aforementioned iterative control methods, with novel and most recent results, specifically, in presence of discrete inputs, discussed in Rawlings and Risbeck (2017) [67].

2.4. Scheduling state-space model

Motivated by process control approaches, Subramanian et al. (2012) [24] proposed a state-space model (Eqs. 5, 11-16) for the chemical production scheduling problem. For brevity, we present the formulation for constant batch-sizes ($\beta_{ij}^{min} = \beta_{ij}^{max} = \beta_{ij}$). There are two distinct features of this model. First, the "complete status" of the plant can be interpreted solely from the variables (states) at that moment in time. This is made possible by *lifting* past actions/inputs (the task start binary variables, W_{ijt}) which have a lagged effect on the "current status" of the plant. Second, observed uncertainties are treated as disturbances, and represented as parameters in the model equations. These two features, together, allow for the model to be kept identical in each online scheduling iteration without any ad-hoc adjustments (due to observation of uncertainty). Thus, the model is in "online ready" form. In addition, due to the use of the state-space formulation, which is popular in process control models, this model also happens to be a very suitable candidate for integration of scheduling and control [68].

To enable lifting of inputs, new task-states (variables) $\bar{W}_{ijt}^n$ are defined. Although, this increases the number of variables in the model, it is matched by an equal increase in the number of equations (the lifting equations, Eqs. 11 and 12). Thus, no new degrees of freedom are introduced. When the task starts, n is zero ($n = 0$, Eq. 11), and when the task finishes, n equals the processing time of the task ($n = \tau_{ij}$). To express task delay and unit breakdown disturbances, new parameters $\hat{Y}_{ijt}^n$ and $\hat{Z}_{ijt}^n$ are defined, respectively. $\hat{Y}_{ijt}^n$, when 1, denotes a delay of δ h in task i during time-period $[t-\delta, t)$, where δ, as defined in Section 2.1.3, is the granularity of the discrete time-grid. $\hat{Z}^n_{ijt}$, when 1, denotes break-down of unit j while executing task i during time-period $[t-\delta, t)$. For ease of presentation, we assume from here on that $\delta = 1$ h.

$$\bar{W}^0_{ijt} = W_{ijt} \quad \forall j, i \in \mathbf{I}_j, t \tag{11}$$

$$\bar{W}^n_{ij(t+1)} = \bar{W}^{n-1}_{ijt} - \hat{Y}^{n-1}_{ijt} + \hat{Y}^n_{ijt} - \hat{Z}^{n-1}_{ijt} \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1, 2, ..., \tau_{ij}\} \tag{12}$$

In the absence of delays or breakdowns, the lifting equations effectively represent the relation: $\bar{W}^n_{ijt} = W_{ij(t-n)}$ $\forall j, i \in \mathbf{I}_j, n$. The lifted variables are defined only till $n = \tau_{ij}$, because a "look-back" beyond that value of n is not needed. The effect of past inputs, for $n > \tau_{ij}$, is already, indirectly, contained in the inventory and backlog variables S_{kt} and BO_{kt}. The lifted states, $\bar{W}^n_{ijt}$, are augmented to the future states (see Fig. 2).

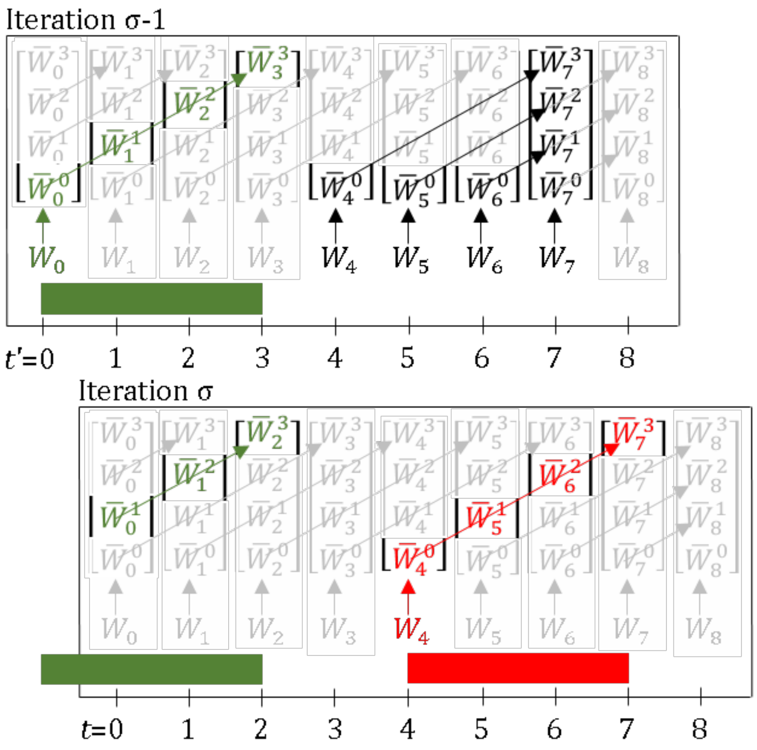

Iteration σ-1
$\bar{W}_0^3$ $\bar{W}_0^2$ $\bar{W}_0^1$ $\bar{W}_0^0$
$\bar{W}_1^3$ $\bar{W}_1^2$ $\bar{W}_1^1$ $\bar{W}_1^0$
$\bar{W}_2^3$ $\bar{W}_2^2$ $\bar{W}_2^1$ $\bar{W}_2^0$
$\bar{W}_3^3$ $\bar{W}_3^2$ $\bar{W}_3^1$ $\bar{W}_3^0$
$\bar{W}_4^3$ $\bar{W}_4^2$ $\bar{W}_4^1$ $\bar{W}_4^0$
$\bar{W}_5^3$ $\bar{W}_5^2$ $\bar{W}_5^1$ $\bar{W}_5^0$
$\bar{W}_6^3$ $\bar{W}_6^2$ $\bar{W}_6^1$ $\bar{W}_6^0$
$\bar{W}_7^3$ $\bar{W}_7^2$ $\bar{W}_7^1$ $\bar{W}_7^0$
$\bar{W}_8^3$ $\bar{W}_8^2$ $\bar{W}_8^1$ $\bar{W}_8^0$
W_0 W_1 W_2 W_3 W_4 W_5 W_6 W_7 W_8
t'=0 1 2 3 4 5 6 7 8
Iteration σ
$\bar{W}_0^3$ $\bar{W}_0^2$ $\bar{W}_0^1$ $\bar{W}_0^0$
$\bar{W}_1^3$ $\bar{W}_1^2$ $\bar{W}_1^1$ $\bar{W}_1^0$
$\bar{W}_2^3$ $\bar{W}_2^2$ $\bar{W}_2^1$ $\bar{W}_2^0$
$\bar{W}_3^3$ $\bar{W}_3^2$ $\bar{W}_3^1$ $\bar{W}_3^0$
$\bar{W}_4^3$ $\bar{W}_4^2$ $\bar{W}_4^1$ $\bar{W}_4^0$
$\bar{W}_5^3$ $\bar{W}_5^2$ $\bar{W}_5^1$ $\bar{W}_5^0$
$\bar{W}_6^3$ $\bar{W}_6^2$ $\bar{W}_6^1$ $\bar{W}_6^0$
$\bar{W}_7^3$ $\bar{W}_7^2$ $\bar{W}_7^1$ $\bar{W}_7^0$
$\bar{W}_8^3$ $\bar{W}_8^2$ $\bar{W}_8^1$ $\bar{W}_8^0$
W_0 W_1 W_2 W_3 W_4 W_5 W_6 W_7 W_8
t=0 1 2 3 4 5 6 7 8

Figure 2. Task-states are shown for two online iterations – numbered $\sigma - 1$ and σ. Each iteration uses its own local time-grid which is reset to start from 0. Here, τ_{ij} for the tasks is assumed to be 3. Lifting of past inputs enables knowing the complete status of the plant by looking at the states (variables) only at that moment in time. In the absence of delays or breakdowns, the lifting equations effectively represent the relation: $\bar{W}^n_{ijt} = W_{ij(t-n)}$ $\forall j, i \in \mathbf{I}_j, n$. Arrows show which variables are equal due to the lifting equations (Eqs. 11-12, with no delays or breakdowns). Variables in green or red have a value of 1, rest have value 0. Information is carried over from one iteration to the next through the update step (Eqs. 17-19).

In the assignment constraint (Eq. 13), parameters $\hat{Y}^{\tau_{ij}}_{ijt}$ and $\hat{Z}^{\tau_{ij}}_{ijt}$ are included, to ensure that the unit appears to be busy, and no new tasks can be started, when there is a delay or breakdown observed at a time when a task is about to finish. Additionally, for multi-period breakdowns, parameter $\hat{Z}^{\tau_{\mathrm{IT},j}}_{\mathrm{IT},jt}$ is made 1, where IT is a fictitious "idle task", with $\tau_{\mathrm{IT},j} = 1$, that keeps the unit busy through the duration of the multi-period breakdown.

$$\sum_{i\in\mathbf{I}_j} W_{ijt} + \sum_{i\in\mathbf{I}_j}\sum_{n=1}^{\tau_{ij}-1} \bar{W}^n_{ijt} + \sum_{i\in\mathbf{I}_j}(\hat{Y}^{\tau_{ij}}_{ijt} + \hat{Z}^{\tau_{ij}}_{ijt}) \le 1 \quad \forall\, j,t \tag{13}$$

In inventory balance (Eq. 14), $\hat{\beta}^C_{ijkt}$ and $\hat{\beta}^P_{ijkt}$ are parameters that denote material handling loss during consumption and production of material k, respectively. When a delay or breakdown is observed at the end of a task, the terms $\hat{Y}^{\tau_{ij}}_{ijt}$ and $\hat{Z}^{\tau_{ij}}_{ijt}$, which are subtracted from $\bar{W}^{\tau_{ij}}_{ijt}$, prevent erroneous multiple counting of the material amount produced by that task.

$$\begin{aligned} S_{k(t+1)} = S_{kt} &+ \sum_j \sum_{i\in\mathbf{I}_j\cap\mathbf{I}^+_k} (\bar{\rho}_{ik}\beta_{ij}(\bar{W}^{\tau_{ij}}_{ijt} - \hat{Y}^{\tau_{ij}}_{ijt} - \hat{Z}^{\tau_{ij}}_{ijt}) + \hat{\beta}^P_{ijkt}) \\ &+ \sum_j \sum_{i\in\mathbf{I}_j\cap\mathbf{I}^-_k} (\rho_{ik}\beta_{ij}W_{ijt} + \hat{\beta}^C_{ijkt}) - V_{kt} + \zeta_{kt} \quad \forall\, k,t \end{aligned} \tag{14}$$

In the backorder balance (Eq. 15), $\hat{\xi}_{kt}$ denotes demand disturbance.

$$BO_{k(t+1)} = BO_{kt} - V_{kt} + \hat{\xi}_{kt} \quad \forall\, k,t \tag{15}$$

Finally, Eq. 16 shows the bounds on the variables present in the model.

$$W_{ijt}, \bar{W}^n_{ijt} \in \{0,1\};\; S_{kt}, BO_{kt}, V_{kt} \ge 0 \tag{16}$$

Next, we describe the online update step, i.e., how information is carried over from one online iteration to the next. Since the scheduling horizon is advanced by 1 h (the model is kept identical), the state at $t = 0$ (initial condition) for the next iteration is matched with the state at $t' = 1$ of the previous iteration. This is shown in Fig. 2, and achieved through the online "update equations" (Eqs. 17-19), in which σ denotes the iteration number. Variables ${}_{\sigma}S_{k(t=0)}$, ${}_{\sigma}BO_{k(t=0)}$, and ${}_{\sigma}\bar{W}^n_{ij(t=0)}$ for $n \ge 1$ which represent lifted task-states, are assigned fixed values through the update step. However, ${}_{\sigma}\bar{W}^0_{ij(t=0)}$, which represents degrees of freedom to start new tasks at $t = 0$, is not fixed. This is identical to how the online updates are performed for the no disturbance case [25]. But, since, here we are dealing with the case where disturbances can be present, the disturbance parameters ($\hat{\xi}_{kt}$, $\hat{Y}^n_{ijt}$, $\hat{Z}^n_{ijt}$, $\hat{\beta}^P_{ijkt}$, and $\hat{\beta}^C_{ijkt}$) are also assigned appropriate values to reflect the observed disturbances. However, these parameters do not participate in the update equations. These influence the prediction of states, for $t \ge 1$, in the online iteration σ.

$${}_{\sigma}\bar{W}^n_{ij(t=0)} = {}_{(\sigma-1)}\bar{W}^n_{ij(t'=1)} \quad \forall j, i \in \mathbf{I}_j, n \in \{1,2,...,\tau_{ij}\} \tag{17}$$

$${}_{\sigma}S_{k(t=0)} = {}_{(\sigma-1)}S_{k(t'=1)} \quad \forall k \tag{18}$$

$${}_{\sigma}BO_{k(t=0)} = {}_{(\sigma-1)}BO_{k(t'=1)} \quad \forall k \tag{19}$$

Figures 3A and 3B, show the evolution of task-states when a 2 h delay is observed right after a task starts and just before a task is about to finish, respectively. The 2 h duration of this multi-period delay is known immediately in iteration σ. However, the model formulation also does allow for representing the observation of consecutive, possibly independent, 1 h single-period delays, one at a time in succeeding iterations. These collectively, in hindsight, appear to be a single multi-period delay, but are actually not.

It is quite evident, that n, now in the presence of delays, loses its physical meaning of denoting how much progress has been made on the task. For example, for the task in Fig. 3A (iteration σ), due to the 2 h delay, the task-states evolve as $\bar{W}^1_{(t=0)} = \bar{W}^1_{(t=1)} = \bar{W}^1_{(t=2)} = \bar{W}^2_{(t=3)} = \bar{W}^3_{(t=4)} = 1$, instead of the more intuitive $\bar{W}^0_{(t=0)} = \bar{W}^0_{(t=1)} = \bar{W}^1_{(t=2)} = \bar{W}^2_{(t=3)} = \bar{W}^3_{(t=4)} = 1$. Similarly, in Fig. 3B (iteration σ), the task-states evolve as $\bar{W}^3_{(t=0)} = \bar{W}^3_{(t=1)} = \bar{W}^3_{(t=2)} = 1$, instead of the more intuitive $\bar{W}^2_{(t=0)} = \bar{W}^2_{(t=1)} = \bar{W}^3_{(t=2)} = 1$. All that can be said now is that when $\bar{W}^0_{ijt} = 1$, the task has just started at time t, and when $\bar{W}^{\tau_{ij}}_{ijt} = 1$ and $\hat{Y}^{\tau_{ij}}_{ijt} = 0$ simultaneously, then the task has finished. As we will show in Section 3.1, we overcome this limitation by introducing a new convention to map observed disturbances to the disturbance parameters, and hence, are able to preserve the physical meaning of n, even when disturbances are present (see Fig. 5).

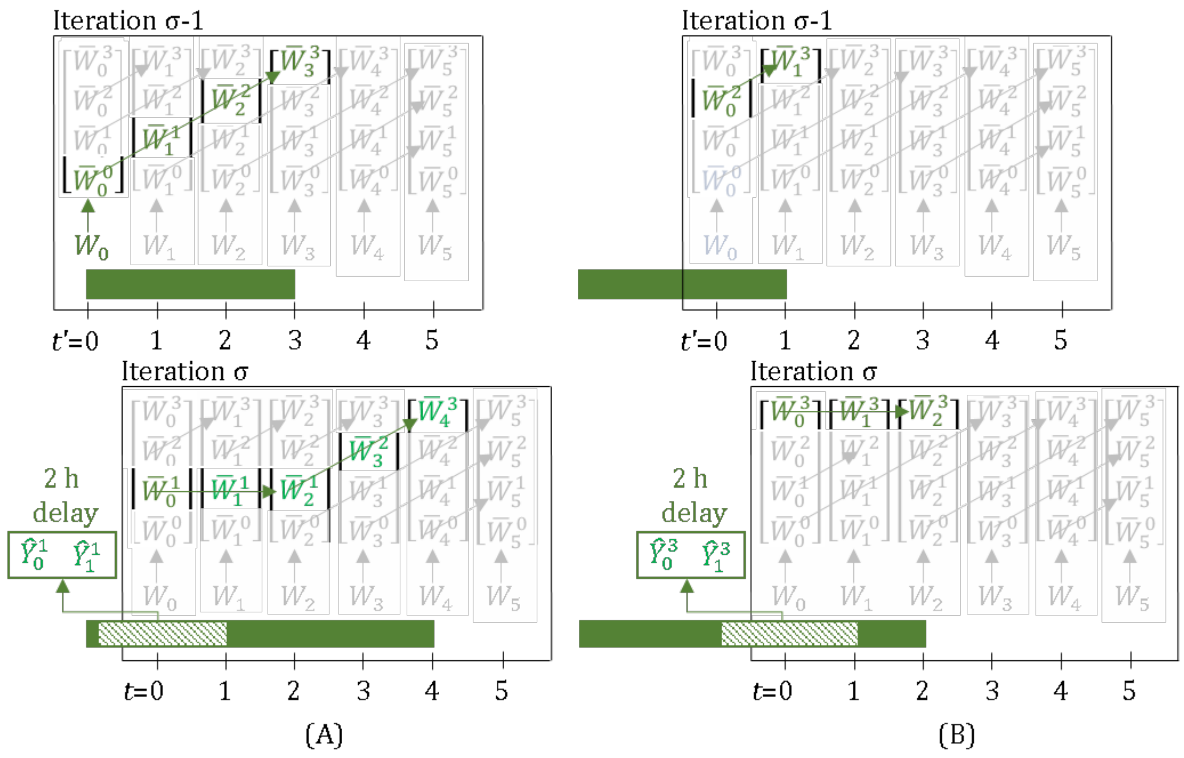


Figure 3. When a 2 h delay is observed, through the lifting equations, $\bar{W}^n_{ijt}$ evolve over the green trajectory, leading to the task correctly finishing 2 h late in iteration σ. Here, τ for the task is 3. Arrows show which variables are enforced as equal by the lifting equations (Eqs. 11-12, with delays present). Variables and parameters in green have a value of 1, rest have value 0. (A) The task now finishes at $t = 4$, instead of at $t = 2$. (B) The task now finishes at $t = 2$, instead of at $t = 0$. Through Eq. 13, the unit is kept busy at $t = 0$ and 1, by the inclusion of the terms $\hat{Y}^{\tau=3}_{(t=0)}$ and $\hat{Y}^{\tau=3}_{(t=1)}$, and hence a new task is prevented from starting at these times. In addition, these terms in Eq. 14, prevent the task's produce from erroneously contributing to inventory ($S_{k,(t=1)}$ and $S_{k,(t=2)}$).

Figures 4A and 4B, show the evolution of task-states when a breakdown just before $t = 0$ is observed and is known to have a 2 h unit downtime (for repairs), right after a task starts and just before a task is about to finish, respectively. Given the observation of breakdown, we would expect intuitively, and unlike what is shown in Figs. 4A and 4B, that none of the task-states $\bar{W}^n_{ijt}$ are active for the green task at $t = 0$. We show how this is achieved through the new model discussed in Section 3.1 (see Fig. 6).

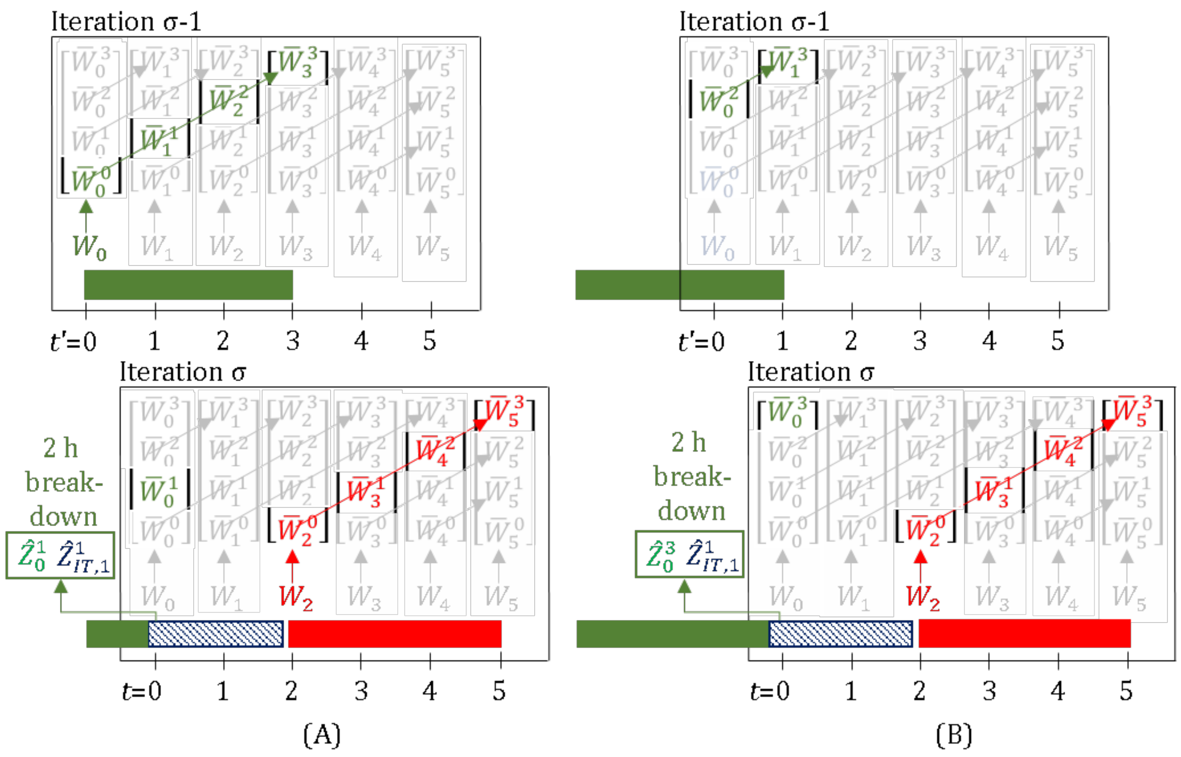


Figure 4. When a breakdown is observed, further evolution of the task-states for the task (the green trajectory), running on the unit that broke, stops. Here, τ for the task is 3 and the unit downtime (blue) is 2 h. Arrows show which variables are enforced as equal by the lifting equations (Eqs. 11-12, with breakdown present). Variables and parameters in green, blue, or red have a value of 1, rest have value 0. The green task is suspended at $t = 0$. A new task (red) can only start at $t = 2$, once the unit downtime is over. (A) Through Eq. 13, the unit is kept busy at $t = 0$ and 1, due to the terms $\bar{W}_{t=0}^{1}$ and $\hat{Z}_{\text{IT},1}^{1}$, respectively. (B) Through Eq. 13, the unit is kept busy at $t = 0$ and 1, due to the terms $\bar{Z}_{t=0}^{\tau=3}$ and $\hat{Z}_{\text{IT},1}^{\tau=1}$, respectively. Additionally, the term $\hat{Z}_{(t=0)}^{\tau=3}$ in Eq. 14, prevents the green task-state ($\bar{W}_{t=0}^{\tau_{ij}=3}$) from erroneously contributing to the inventory.

3. Modeling generalizations

In Section 3.1, we present a new state-space model formulation that differs, from the state-space model of Subramanian et al. (2012) [24], in the convention that is followed for mapping observed disturbances to the disturbance parameters. Although both models are accurate, this new convention ensures that the task-states, in the presence of disturbances, follow a more intuitive notation. Specifically, the meaning of n as the progress of a task, is maintained. In addition, we define several new parameters to systematically account for disturbances.

In Section 3.2, we show how to handle fractional delays and unit downtimes (due to unit breakdowns). In Section 3.3, we expand the scope of the model to account for variable batch-sizes. Thereafter, in Sections 3.4-3.9 we present generalizations that can be applied to the state-space model, one at a time. Afterwards, in Section 4, we present the final model equations with all generalizations present simultaneously. As we will see in that section, for all the generalizations to work in the presence of each other, a few more modifications are necessary.

3.1. New basic formulation

The new state-space model relies on a comprehensive update step of the task-states, in between the online iterations, to promptly reflect the delays and breakdowns in the task-states. The inventory and backorder update stay the same (Eqs. 18-19) as in the model of Subramanian et al. (2012) [24]. The task-states update is modified from Eq. 17 to Eqs. 20-21.

$$
\begin{aligned}
{}_{\sigma}\bar{W}^{n}_{ij(t=0)} &= {}_{(\sigma-1)}\bar{W}^{n-1}_{ij(t'=0)} - \dot{Y}^{n-1}_{ij} + \dot{Y}^{n}_{ij} - \dot{Z}^{n-1}_{ij} \quad \forall j, i \in \mathbf{I}_j, n \in \{1, 2, ..., \tau_{ij}\} && (20)\\
{}_{\sigma}X_{ij(t=0)} &= \dot{Y}^{0}_{ij} \quad \forall j, i \in \mathbf{I}_j && (21)
\end{aligned}
$$

The parameters $\dot{Y}^{n}_{ij}$ which, if 1, represent a 1 h delay in task with progress status n. Note the dot $(\cdot)$ instead of the hat $(\wedge)$ on the symbols of these parameters. Since these parameters are exclusively for the update step, and do not directly participate in the optimization model, these need not be indexed by time – neither t' (iteration $\sigma - 1$) nor t (iteration σ). Similarly, $\dot{Z}^{n}_{ij}$ denotes a breakdown of unit j on which task i with progress status n was running. X_{ijt} is a new binary variable, defined for all time-points, which, when 1, captures the information about delays in a task with progress status $n = 0$, i.e., when the task gets delayed right after it starts. The use of this variable, in Eqs. 22-23, will become clear when we discuss the optimization model. We also define a new parameter $\hat{\Lambda}_{jt}$, which, when 1, denotes the unit is unavailable for the time-period $[t, t+1)$. This parameter participates in Eq. 25.

For a multi-period delay of ϕ h, in task i on unit j, in addition to $\dot{Y}^{n}_{ij}$, parameters $\hat{Y}^{n}_{ijt}$ are activated for $t = 0, t = 1, ..., t = \phi - 2$. For unit breakdowns with downtime duration of ϕ h, in addition to $\dot{Z}^{n}_{ij}$, parameters, $\hat{\Lambda}_{jt}$ are activated for $t = 0, t = 1, ..., t = \phi - 1$. Thus, single-period delays do not result in activation of any $\hat{Y}^{n}_{ijt}$ parameters, but single-period breakdowns require activation of $\hat{\Lambda}_{j(t=0)}$.

Having described the update step, we now describe the optimization model. In this model, the lifting equations consist of Eqs. 22-24.

$$
\begin{aligned}
X_{ij(t+1)} &= \hat{Y}^{0}_{ijt} \quad \forall j, i \in \mathbf{I}_j, t && (22)\\
\bar{W}^{0}_{ijt} &= W_{ijt} + X_{ijt} \quad \forall j, i \in \mathbf{I}_j, t && (23)\\
\bar{W}^{n}_{ij(t+1)} &= \bar{W}^{n-1}_{ijt} - \hat{Y}^{n-1}_{ijt} + \hat{Y}^{n}_{ijt} \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1, 2, ..., \tau_{ij}\} && (24)
\end{aligned}
$$

When there is a ϕ h multi-period delay in a task with progress $n = 0$, the update step assigns $X_{ij(t=0)} = 1$ and $\hat{Y}^{0}_{ijt} = 1 \ \forall t \in \{0, 1, ..., \phi - 2\}$. This ensures that $\bar{W}^{0}_{ijt}$ stays activated for next (ϕ-1) h, but with $W_{ijt} = 0$, i.e., the task is not erroneously interpreted as a new task start. If there are no delays, then, through Eqs. 21-22, $X_{ijt} = 0$ and any new task that starts with $W_{ijt} = 1$, through Eq. 23, results in $\bar{W}^{0}_{ijt} = 1$. Equation 23 is a constraint that we impose on the inputs (W_{ijt}) given the states (X_{ijt} and $\bar{W}^{0}_{ijt}$), and if needed can be converted to inequality form through use of Eq. 9. Variables X_{ijt} are either fixed ($t = 0$) by the update step or are equated to the delay parameters in the optimization model, hence, can be declared as free variables with no explicit bounds.

The assignment constraint (Eq. 25) includes the parameter $\hat{\Lambda}_{jt}$ to account for unit downtime. Additionally, it contains the variable $\bar{W}^{0}_{ijt}$ on the left-hand side, and not variable W_{ijt}, to correctly account for the unit being busy, specifically, when a delay in a task with progress $n = 0$ is observed.

$$
\sum_{i \in \mathbf{I}_j} \bar{W}^{0}_{ijt} + \sum_{i \in \mathbf{I}_j} \sum_{n=1}^{\tau_{ij}-1} \bar{W}^{n}_{ijt} \leq 1 - \hat{\Lambda}_{jt} \quad \forall\, j, t \qquad (25)
$$

The inventory balance, Eq. 26, in contrast to Eq. 14, does not require any corrective delay or breakdown terms. This is because, for any task, the states W_{ijt} (task-start) and $\bar{W}_{ijt}^{\tau_{ij}}$ (task-end), even if delays or breakdowns are observed, are active only at most once.

$$S_{k(t+1)} = S_{kt} + \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^+} (\bar{\rho}_{ik} \beta_{ij} \bar{W}_{ijt}^{\tau_{ij}} + \hat{\beta}_{ijkt}^P) + \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^-} (\rho_{ik} \beta_{ij} W_{ijt} + \hat{\beta}_{ijkt}^C) - V_{kt} + \zeta_{kt} \quad \forall\, k, t \tag{26}$$

The complete optimization model consists of Eqs. 5, 15-16, and 22-26. Figures 5 and 6, respectively, show the evolution of task-states when delays or breakdowns are observed.

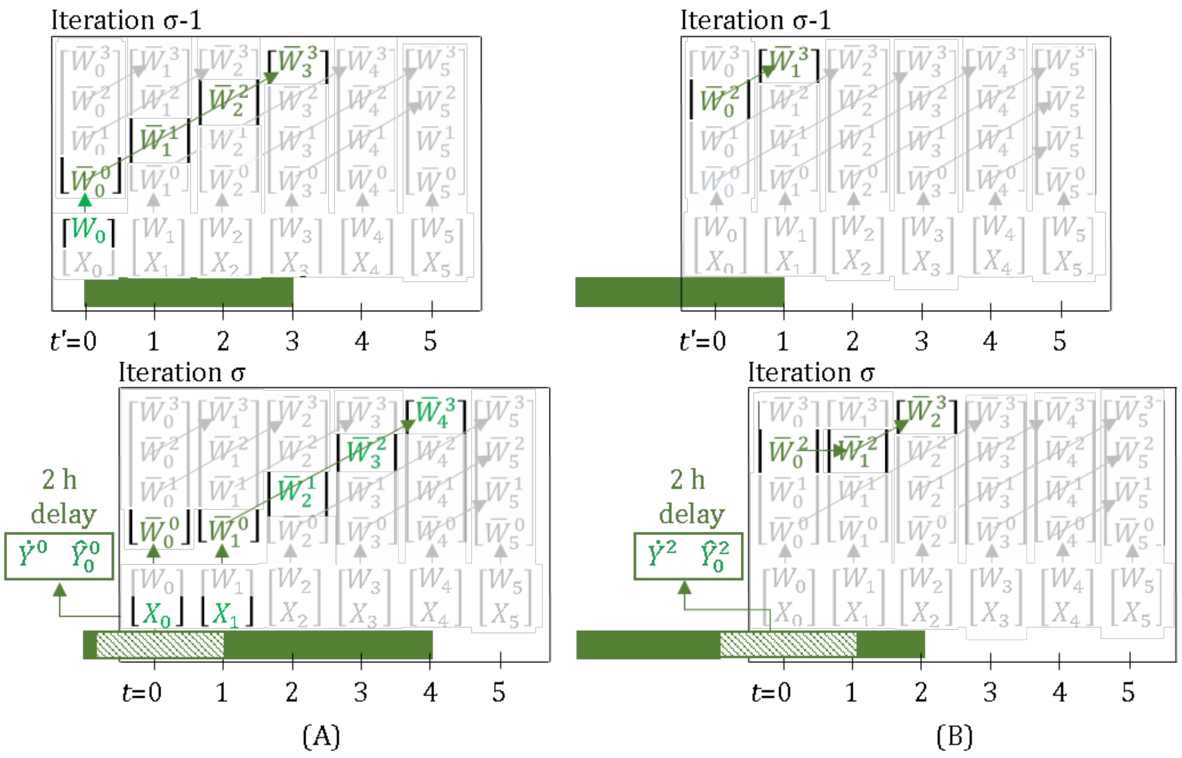


Figure 5. When a 2 h delay is observed, through the lifting equations, $\bar{W}_{ijt}^n$ evolve over the green trajectory, leading to the task correctly finishing 2 h late in iteration σ. Here, τ for the task is 3. Arrows show which variables are enforced as equal by the lifting equations (Eqs. 22-24, with delays present). Variables and parameters in green have a value of 1, rest have value 0. (A) The task now finishes at $t = 4$, instead of at $t = 2$. Due to the update step, parameters $\dot{Y}$, $\hat{Y}_0^0$ and variable X_0 are 1, hence, in iteration σ, due to the optimization model, $\bar{W}_0^0$, X_1, and $\bar{W}_1^0$ are also 1. (B) The task now finishes at $t = 2$, instead of at $t = 0$. The update step ensures that the true progress, n, of the task is reflected in the task-states at $t = 0$, i.e. $n = 2$.

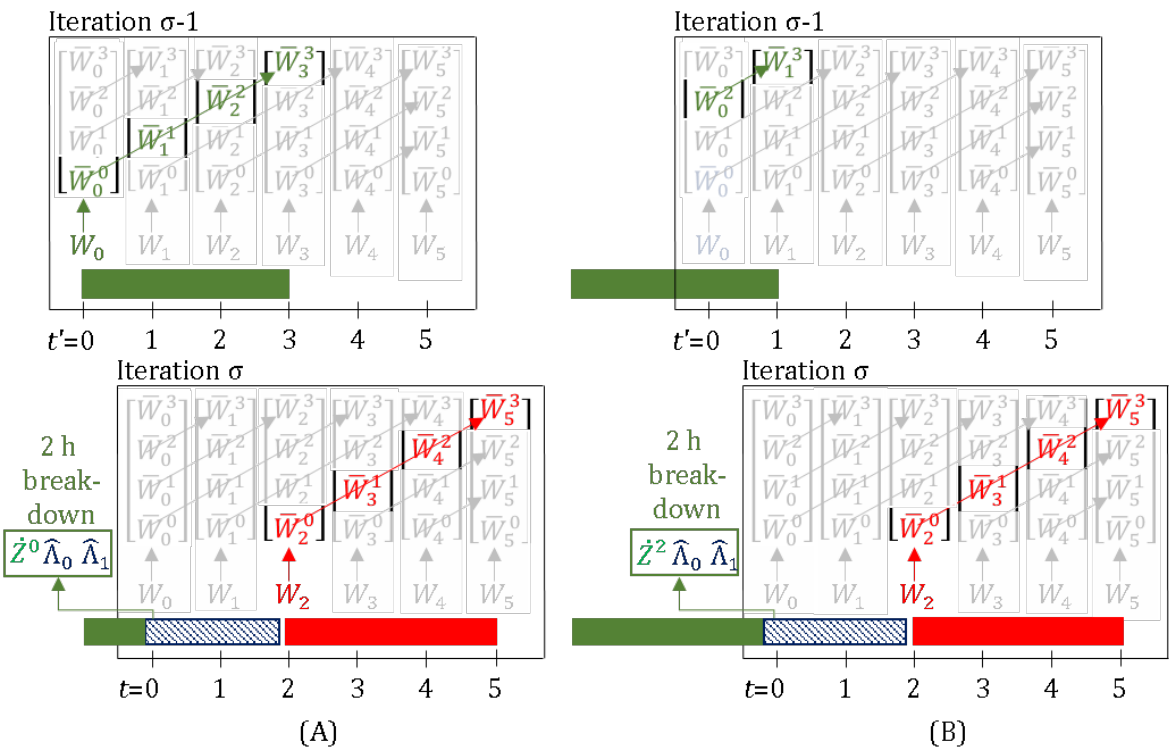

Iteration σ-1
Iteration σ
2 h break-down
$\dot{Z}^0 \hat{\Lambda}_0 \hat{\Lambda}_1$
$\dot{Z}^2 \hat{\Lambda}_0 \hat{\Lambda}_1$
t'=0 1 2 3 4 5
t=0 1 2 3 4 5
(A)
(B)

Figure 6. When a breakdown is observed, further evolution of the task-states for the task (the green trajectory), running on the unit that broke, stops. Here, τ for the task is 3 and the unit downtime (blue) is 2 h. Arrows show which variables are enforced as equal by the lifting equations (Eqs. 22-24). Variables and parameters in green, blue, or red have a value of 1, rest have value 0. The green task is suspended at $t = 0$. A new task (red) can only start at $t = 2$, once the unit downtime is over. Through Eq. 25, the unit is kept busy at $t = 0$ and 1, by the inclusion of the terms $\hat{\Lambda}_{t=0}$ and $\hat{\Lambda}_{t=1}$. (A) The parameter $\dot{Z}^0$, through Eq. 20, prevents the task-state from evolving from ${}_{(\sigma-1)}\bar{W}_0^0$ to ${}_{\sigma}\bar{W}_0^1$. (B) The parameter $\dot{Z}^2$, through Eq. 20, prevents the task-state from evolving from ${}_{(\sigma-1)}\bar{W}_0^2$ to ${}_{\sigma}\bar{W}_0^3$.

3.2. *Fractional delays and unit downtimes*

In Figs. 5 and 6, we showed the cases where delays and unit downtime are integer multiples of time-grid spacing δ. Additionally, the unit breakdown was assumed to take place at almost the time-point t, i.e., very close to an integer multiple of δ. However, if δ is not very small, then these assumptions may not be good. Given any fractional delays (π_{delay}), downtimes (π_{down}), or unit breakdown time (π_{break}), we need an appropriate scheme for the (online iterations) update step, to ensure realistic rounding of these to integer values, so as to keep the task-finish and unit-availability times, in sync with the discrete time-grid. A single task can have multiple separate delays, hence, we index the delay time with index r (recurrence), i.e., π^r_{delay}. A breakdown, however, can occur only once, at π_{break}, following which, the unit downtime, π_{down}, starts.

For the first delay, a rounded up value is applied in the update steps, i.e. the delay is assumed to be $\lceil \pi^1_{delay}/\delta \rceil$. For every additional ψ^{th} delay, the difference, $\phi = \lceil (\sum_{r=1}^{\psi} \pi^r_{delay})/\delta \rceil - \lceil (\sum_{r=1}^{\psi-1} \pi^r_{delay})/\delta \rceil$, dictates how much additional, integer ϕ, delay is applied in the update steps. Figure 7A shows a numerical example for fractional delays.

When a unit breaks down, the parameter $\dot{Z}^n$ is always activated, so as to suspend the running task. The key challenge is to identify, for how many next time-points the unit would be unavailable. This dictates, if, and how many, $\hat{\Lambda}_t$ parameters are activated. This is done as follows. On breakdown, the unit becomes unavailable from π_{break} to $\pi_{break} + \pi_{down}$ (in iteration σ, $\pi_{break} < 0$). Hence, all

$\hat{\Lambda}_t$ that span integer multiple of δ, $t \in (\pi_{break}, \pi_{break} + \pi_{down}]$ are activated. This also means, if $(\pi_{break} - \lfloor \pi_{break}/\delta \rfloor \delta + \pi_{down}) < \delta$, none of the $\hat{\Lambda}_t$ are activated, i.e., the unit breaks down and comes back online before the immediate next time-point. This is illustrated in Fig. 7B.

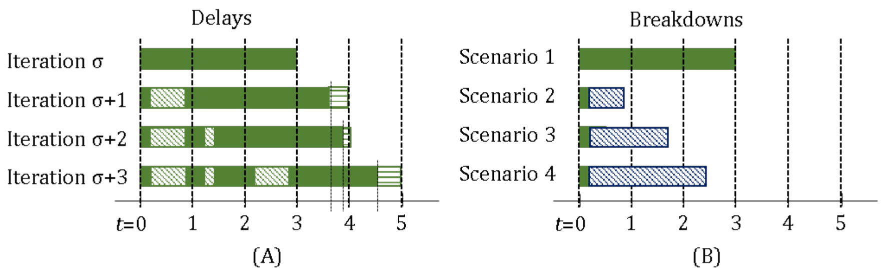


Figure 7. The task has a nominal processing time $\tau = 3$. Time-grid spacing is $\delta = 1$ h. Here, for ease of discussion, the time-grid is global, i.e., it is not reset for each iteration. (A) A delay (shown as oblique green pattern) of 0.66 h in time-period (0,1) is observed. Since, $\lceil 0.66 \rceil = 1$, a delay of 1 h is applied at $t = 1$. This would ensure that the task finish is aligned with $t = 4$, even if the task actually ends at $t = 3.66$. The horizontal green pattern represents the fictitious extra task runtime to align with the discrete time-grid. Next, another delay of 0.2 h is observed in time-period (1,2). Since, $\lceil 0.66 + 0.2 \rceil - \lceil 0.66 \rceil = 0$, no additional delay is applied in the update steps. This makes sense, because, the task finishes at $t = 3.86$ in reality. Since, the previous delay was applied as 1 h, the task is now still thought to finish at $t = 4$, in alignment with the time-grid. Finally, when another delay of 0.66 h is observed in time-period (2,3). Since, $\lceil 0.66 + 0.2 + 0.66 \rceil - \lceil 0.66 + 0.2 \rceil = 1$, a 1 h delay is applied in the update step. This correctly ensures that the task is now thought to end at $t = 5$, which is the round up of the true end time of $t = 4.52$ h. (B) A break-down is observed at $t = 0.2$. Thus, $\dot{Z}^0 = 1$ for the update step between the iterations starting at $t = 0$ and $t = 1$. If the downtime (blue) is 0.66 h, then the unit actually becomes available at $t = 0.86$. Thus, $\hat{\Lambda}_{(t=1)}$ is not activated. This is indeed the case from our mathematical procedure as well, since, $t \in (0.2, 0.86]$ does not include any integer time-point. If $\pi_{down} = 1.5$ h, then $t \in (0.2, 1.7]$ does span $t = 1$, and consequently $\hat{\Lambda}_{(t=1)} = 1$. Finally, if $\pi_{down} = 2.25$ h, then $t \in (0.2, 2.45]$ spans $t = 1$ and $t = 2$, which results in $\hat{\Lambda}_{(t=1)} = 1$ and $\hat{\Lambda}_{(t=2)} = 1$.

3.3. Variable batch-sizes

To account for variable batch-sizes, we define variables, B_{ijt} which denotes the batch-size of the task that just starts, $\hat{B}^n_{ijt}$ for lifted task batch-size states, and ${}^B X_{ijt}$ to represent batch-size of task that is delayed with progress status $n = 0$. We define parameters ${}^B\dot{Y}_{ij}$ and ${}^B\dot{Z}^n_{ij}$ that participate in the update steps and these denote the batch-size of the task delayed and suspended due to unit break-down, respectively. Further, we define parameter ${}^B\hat{Y}^n_{ijt}$ for the optimization model.

The additional update steps, Eqs. 27-28, due to variable batch-sizes, are as follows:

$$ {}_{\sigma}\bar{B}^n_{ij(t=0)} = {}_{(\sigma-1)}\bar{B}^{n-1}_{ij(t'=0)} - {}^B\dot{Y}^{n-1}_{ij} + {}^B\dot{Y}^n_{ij} - {}^B\dot{Z}^{n-1}_{ij} \quad \forall j, i \in \mathbf{I}_j, n \in \{1, 2, ..., \tau_{ij}\} \tag{27} $$

$$ {}^B_{\sigma}X_{ij(t=0)} = {}^B\dot{Y}^0_{ij} \quad \forall j, i \in \mathbf{I}_j \tag{28} $$

The optimization model now requires Eqs. 29-31 for lifting the batch-size:

$$ {}^B X_{ij(t+1)} = {}^B\hat{Y}^0_{ijt} \quad \forall j, i \in \mathbf{I}_j, t \tag{29} $$

$$ \bar{B}^0_{ijt} = B_{ijt} + {}^B X_{ijt} \quad \forall j, i \in \mathbf{I}_j, t \tag{30} $$

$$ \bar{B}^n_{ij(t+1)} = \bar{B}^{n-1}_{ijt} - {}^B\hat{Y}^{n-1}_{ijt} + {}^B\hat{Y}^n_{ijt} \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1, 2, ..., \tau_{ij}\} \tag{31} $$

It might appear, in Eq. 30, that when ${}^{B}X_{ijt} > 0$, nothing prevents B_{ijt} from also erroneously taking on a positive value. This was not an issue in Eq. 23 because the W_{ijt} and $\bar{W}^{0}_{ijt}$ variables there were binary. However, Eq. 2 (repeated below) ensures that B_{ijt} can only take a non-zero value when $W_{ijt} = 1$. Since, through Eq. 23, $W_{ijt} = 0$, whenever $X_{ijt} = 1$, B_{ijt} also takes value 0. The update steps ensure that X_{ijt} and ${}^{B}X_{ijt}$ can only be non-zero simultaneously.

$$\beta^{min}_{ij} W_{ijt} \leq B_{ijt} \leq \beta^{max}_{ij} W_{ijt} \quad \forall\, j, i \in \mathbf{I}_j, t \tag{2}$$

The inventory balance (Eq. 32) now incorporates the new batch-size variables, B_{ijt} and $\bar{B}^{n}_{ijt}$, rather than the task-state binary variables (W_{ijt} and $\bar{W}^{n}_{ijt}$) which was the case in Eq. 26.

$$\begin{aligned} S_{k(t+1)} = S_{kt} &+ \sum_{j} \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^{+}_{k}} (\bar{\rho}_{ik} \bar{B}^{\tau_{ij}}_{ijt} + \hat{\beta}^{P}_{ijkt}) \\ &+ \sum_{j} \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^{-}_{k}} (\rho_{ik} B_{ijt} + \hat{\beta}^{C}_{ijkt}) - V_{kt} + \zeta_{kt} \quad \forall\, k, t \end{aligned} \tag{32}$$

Finally, the variable bounds are as follows:

$$W_{ijt}, \bar{W}^{n}_{ijt} \in \{0, 1\}; B_{ijt}, \bar{B}^{n}_{ijt}, S_{kt}, BO_{kt}, V_{kt} \geq 0 \tag{33}$$

The update step comprises of Eqs. 18-21 and 27-28, and the optimization model consists of Eqs. 2,5, 15, 22-25 and 29-33.

Remark: In principle, we can completely avoid defining the new parameters ${}^{B}\dot{Y}_{ij}$, ${}^{B}\dot{Z}^{n}_{ij}$, ${}^{B}\hat{Y}_{ijt}$, and variable ${}^{B}X_{ijt}$ by reformulating Eqs. 27-31, so as to only use parameters $\dot{Y}_{ij}$, $\dot{Z}^{n}_{ij}$, $\hat{Y}_{ijt}$, and variable X_{ijt}. For example, Eq. 31 can be reformulated to Eq. 34.

$$\bar{B}^{n}_{ij(t+1)} = \bar{B}^{n-1}_{ijt}(1 - \hat{Y}^{n-1}_{ijt} + \hat{Y}^{n}_{ijt}) \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1, 2, ..., \tau_{ij}\} \tag{34}$$

Since, Eq. 34 entails the multiplication of variables with parameters, by itself, it is an acceptable alternate linear formulation. However, when *task termination* is allowed as a scheduling decision (Section 3.7), this reformulation results in bi-linear terms which are undesirable. Thus, we indeed define the new parameters ${}^{B}\dot{Y}_{ij}$, ${}^{B}\dot{Z}^{n}_{ij}$, ${}^{B}\hat{Y}_{ijt}$, and variable ${}^{B}X_{ijt}$, and use Eqs. 27-31 in their native form without the simplifying reformulation discussed in this remark.

3.4. Robust scheduling: batch-sizes

In many applications, it can be prudent to schedule batches bigger than what are needed to just satisfy the nominal demand. This can be, for example, due to the possibility of seeing a demand spike, or to pro-actively compensate for typical material handling losses when a batch finishes. To do so, the parameter $\bar{\rho}_{ik}$ in material inventory balance (Eq. 35b) can be substituted by a scaled down value ($\bar{\rho}^{r}_{ik}$), where $\bar{\rho}^{r}_{ik} < \bar{\rho}_{ik}$. This results in bigger batches starting, since the model now under-predicts the yield of materials from any given batch-size. In order to, however, correctly account for the actual inventory resulting from the finishing of a task, the nominal value of $\bar{\rho}_{ik}$ is used at $t = 0$, along with any yield-loss or material handling loss disturbance (Eq. 35a). As it can be seen in Fig. 8, which is a simple illustration of this modeling generalization, as the iterations progress, a task-finish-state eventually hits $t = 0$, yielding the large yield proportionate to the true (nominal) value of $\bar{\rho}_{ik}$. Now, if there are any material handling losses ($\hat{\beta}^{P}_{ijk(t=0)}$), they can be subtracted from the true yield (in Eq. 35a). It is worth noting that, although $\hat{\beta}^{P}_{ijkt}$ and $\hat{\beta}^{C}_{ijkt}$ are defined for all time-points, they are possibly active only at $t = 0$, if the corresponding uncertainty is observed. Hence, these parameters can be, in principle, dropped from Eq. 35b.

$$S_{k(t+1)} = S_{kt} + \sum_j \sum_{i\in \mathbf{I}_j \cap \mathbf{I}_k^+} (\bar{\rho}_{ik} \bar{B}_{ijt}^{\tau_{ij}} + \hat{\beta}^P_{ijkt}) + \sum_j \sum_{i\in \mathbf{I}_j \cap \mathbf{I}_k^-} (\rho_{ik} B_{ijt} + \hat{\beta}^C_{ijkt}) - V_{kt} + \zeta_{kt} \quad \forall\, k, t \in \{0\} \tag{35a}$$

$$S_{k(t+1)} = S_{kt} + \sum_j \sum_{i\in \mathbf{I}_j \cap \mathbf{I}_k^+} (\bar{\rho}^r_{ik} \bar{B}_{ijt}^{\tau_{ij}} + \hat{\beta}^P_{ijkt}) + \sum_j \sum_{i\in \mathbf{I}_j \cap \mathbf{I}_k^-} (\rho_{ik} B_{ijt} + \hat{\beta}^C_{ijkt}) - V_{kt} + \zeta_{kt} \quad \forall\, k, t \in \{1,2,3,...\} \tag{35b}$$

We can write the above two equations, compactly together, as follows:

$$S_{k(t+1)} = S_{kt} + \sum_j \sum_{i\in \mathbf{I}_j \cap \mathbf{I}_k^+} (\theta^{\bar{\rho}}_{ikt} \bar{B}_{ijt}^{\tau_{ij}} + \hat{\beta}^P_{ijkt}) + \sum_j \sum_{i\in \mathbf{I}_j \cap \mathbf{I}_k^-} (\rho_{ik} B_{ijt} + \hat{\beta}^C_{ijkt}) - V_{kt} + \zeta_{kt} \quad \forall\, k, t \tag{36}$$

, where

$$\theta^{\bar{\rho}}_{ikt} = \begin{cases} \bar{\rho}_{ik} & \forall k, t = \{0\}; \\ \bar{\rho}^r_{ik} & \forall k, t = \{1,2,3,...\} \end{cases} \tag{37}$$

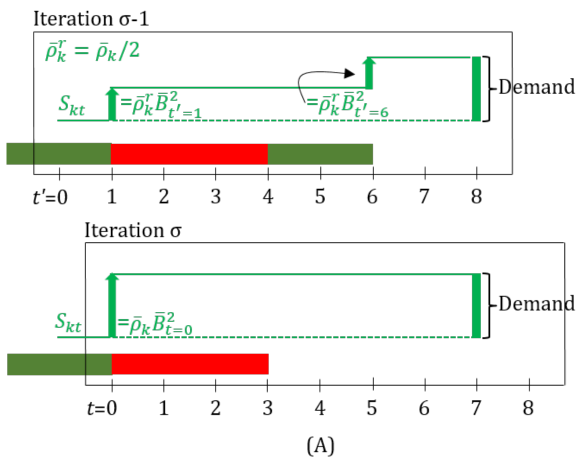
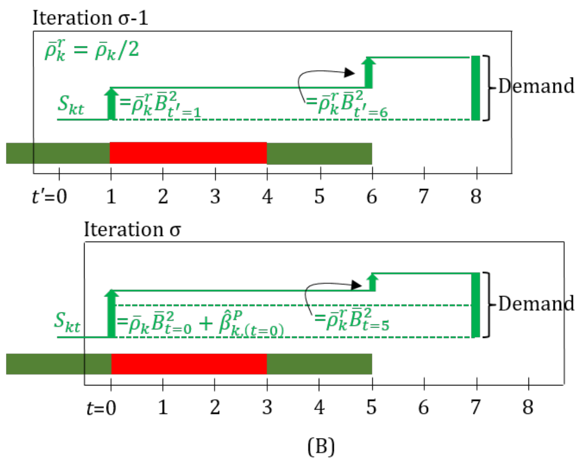

Iteration σ-1
$\bar{\rho}^r_k = \bar{\rho}_k/2$
S_{kt}
$=\bar{\rho}^r_k \bar{B}^2_{t'=1}$
$=\bar{\rho}^r_k \bar{B}^2_{t'=6}$
Demand
t'=0 1 2 3 4 5 6 7 8
Iteration σ
$=\bar{\rho}_k \bar{B}^2_{t=0}$
t=0 1 2 3 4 5 6 7 8
(A)

Iteration σ-1
$\bar{\rho}^r_k = \bar{\rho}_k/2$
S_{kt}
$=\bar{\rho}^r_k \bar{B}^2_{t'=1}$
$=\bar{\rho}^r_k \bar{B}^2_{t'=6}$
Demand
t'=0 1 2 3 4 5 6 7 8
Iteration σ
$=\bar{\rho}_k \bar{B}^2_{t=0} + \beta^P_{k,(t=0)}$
$=\bar{\rho}^r_k \bar{B}^2_{t=5}$
t=0 1 2 3 4 5 6 7 8
(B)

Figure 8. In iteration $\sigma - 1$, the green task ($\tau = 2$) with a "large batch" is finishing at $t' = 1$, but, due to the use of $\bar{\rho}^r_{k'}$, is anticipated to produce only half of what the demand is. Thus, another identical green task is scheduled to start at $t' = 4$, to satisfy the demand. With the use of $\bar{\rho}^r_k$, if the demand could be still be satisfied with a single "large" batch, a second batch wouldn't be scheduled. In iteration σ, the earlier green task yields a large amount of material, in line with its large batch-size due to the true value of $\bar{\rho}_{ik}$ used at $t = 0$. (A) Since, here, there was no material handling loss, thus the second green task need not run now, as the demand was satisfied by the first batch itself. (B) Although the green task results in a large yield at $t = 0$, a small material handling loss ($\beta^P_{k(t=0)} < 0$) at $t = 0$, requires the second green task to still be scheduled in order to meet the demand, but now with a smaller batch-size. If there are no further material handling losses, there would be a small excess inventory of material produced by the second green task, since its batch-size was decided assuming the yield to be lower ($\bar{\rho}^r_k$) but would in reality by higher ($\bar{\rho}_k$).

3.5. Robust scheduling: processing times

Uncertainty in the processing times is very common in scheduling [69]. A popular approach to proactively manage this uncertainty is to *robustify* the schedule by adding a delay buffer to each task's processing time [70]. Once this robust schedule has been computed, it is advantageous to adjust it online, by taking into account the feedback on actual finish times of the tasks [71]. In discrete-time models, this has been typically done using ad-hoc adjustments in between the online iterations. To the best knowledge of the authors, there is not yet a systematic way to be able to naturally handle this adjustment within an optimization model.

We show here, how we can extend the state-space model to produce robust schedules, from a processing time point of view, and yet seamlessly allow for tasks to finish, after they have been running for their nominal processing times plus the delays. We define a new parameter τ^r_{ij}, which denotes the conservative processing time of the tasks ($\tau^r_{ij} > \tau_{ij}$). Thereafter, we modify the lifting (Eqs. 24 modified to Eqs. 38a- 38b and Eq. 31 modified to Eqs. 39a-39b), assignment (Eq. 25 modified to Eqs. 40a-40b), and inventory balance equations (Eq. 32 modified to Eqs. 41a-41b), such that the nominal value of processing times (τ_{ij}) is employed at $t = 0$, and the conservative value (τ^r_{ij}) is employed for $t > 0$. No other model or update equations are modified. An illustration is shown in Fig. 9.

$$\bar{W}^n_{ij(t+1)} = \bar{W}^{n-1}_{ijt} - \hat{Y}^{n-1}_{ijt} + \hat{Y}^n_{ijt} \quad \forall j, i \in \mathbf{I}_j, n \in \{1,2,...,\tau_{ij}\}, t \in \{0\} \tag{38a}$$

$$\bar{W}^n_{ij(t+1)} = \bar{W}^{n-1}_{ijt} - \hat{Y}^{n-1}_{ijt} + \hat{Y}^n_{ijt} \quad \forall j, i \in \mathbf{I}_j, n \in \{1,2,...,\tau^r_{ij}\}, t \in \{1,2,3,...\} \tag{38b}$$

$$\bar{B}^n_{ij(t+1)} = \bar{B}^{n-1}_{ijt} - {}^B\hat{Y}^{n-1}_{ijt} + {}^B\hat{Y}^n_{ijt} \quad \forall j, i \in \mathbf{I}_j, n \in \{1,2,...,\tau_{ij}\}, t \in \{0\} \tag{39a}$$

$$\bar{B}^n_{ij(t+1)} = \bar{B}^{n-1}_{ijt} - {}^B\hat{Y}^{n-1}_{ijt} + {}^B\hat{Y}^n_{ijt} \quad \forall j, i \in \mathbf{I}_j, n \in \{1,2,...,\tau^r_{ij}\}, t \in \{1,2,3,...\} \tag{39b}$$

$$\sum_{i \in \mathbf{I}_j} \bar{W}^0_{ijt} + \sum_{i \in \mathbf{I}_j} \sum_{n=0}^{\tau_{ij}-1} \bar{W}^n_{ijt} \le 1 - \hat{\Lambda}_{jt} \quad \forall\, j, t \in \{0\} \tag{40a}$$

$$\sum_{i \in \mathbf{I}_j} \bar{W}^0_{ijt} + \sum_{i \in \mathbf{I}_j} \sum_{n=0}^{\tau^r_{ij}-1} \bar{W}^n_{ijt} \le 1 - \hat{\Lambda}_{jt} \quad \forall\, j, t \in \{1,2,3,...\} \tag{40b}$$

$$\begin{aligned} S_{k(t+1)} = S_{kt} &+ \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^+_k} (\bar{\rho}_{ik} \bar{B}^{\tau_{ij}}_{ijt} + \hat{\beta}^P_{ijkt}) \\ &+ \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^-_k} (\rho_{ik} B_{ijt} + \hat{\beta}^C_{ijkt}) - V_{kt} + \zeta_{kt} \quad \forall\, k, t \in \{0\} \end{aligned} \tag{41a}$$

$$\begin{aligned} S_{k(t+1)} = S_{kt} &+ \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^+_k} (\bar{\rho}_{ik} \bar{B}^{\tau^r_{ij}}_{ijt} + \hat{\beta}^P_{ijkt}) \\ &+ \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^-_k} (\rho_{ik} B_{ijt} + \hat{\beta}^C_{ijkt}) - V_{kt} + \zeta_{kt} \quad \forall\, k, t \in \{1,2,3,...\} \end{aligned} \tag{41b}$$

The lifting equations, Eqs. 38b and 39b, contain variables $W^n_{ij(t=1)}$, $B^n_{ij(t=1)}$ $\forall j, i \in \mathbf{I}_j, n \in \{\tau_{ij} + 1, \tau_{ij} + 2, ..., \tau^r_{ij}\}$, which are not coupled back to any variables at $t = 0$. These variables must be fixed to zero, otherwise, the optimization can assign these variables a spurious value so as to erroneously generate inventory (in Eq. 41b through variables $B^{\tau^r_{ij}}_{ijt}$ $\forall j, i \in \mathbf{I}_j, t \in \{1,2,...,\tau^r_{ij} - \tau_{ij}\}$).

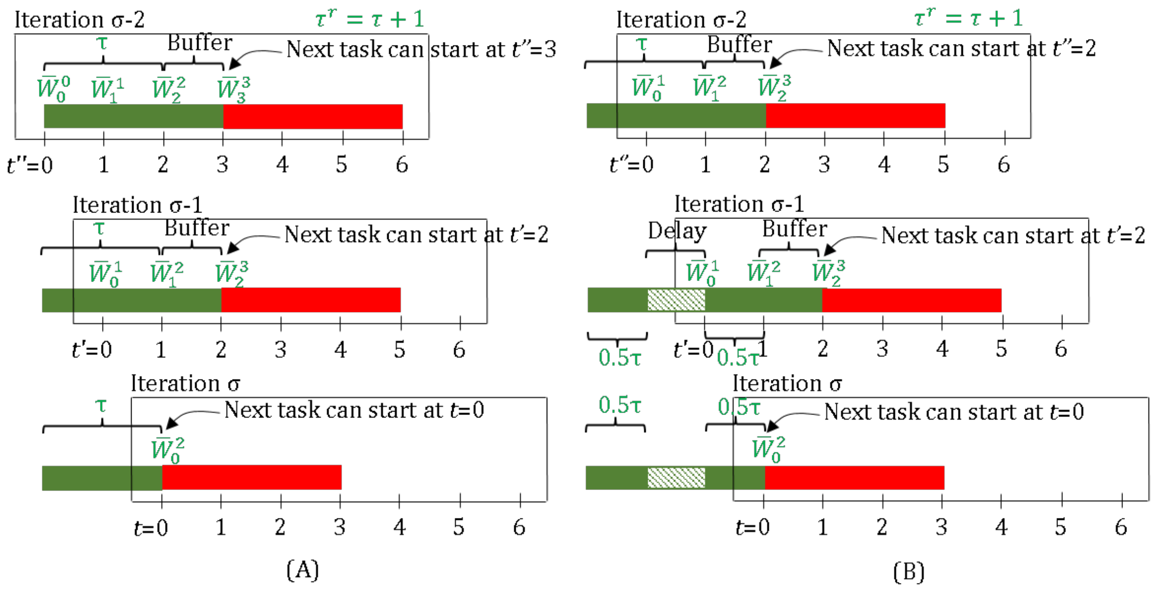


Figure 9. For the green task ($\tau = 2$), a delay buffer of 1 h is chosen, i.e., $\tau^r_{ij} = \tau_{ij} + 1$. (A) In iteration $\sigma - 1$, the model predicts that the green task will finish at $t' = 2$. In the next iteration σ, when there is no delay in the green task, having run for 2 h, it correctly finishes at $t = 0$. No ad-hoc model changes are required in between the iterations to make this possible. Consequently, to take advantage of this nominal finish time of the green task, the red task, now starts 1 h earlier than what it was scheduled for in the previous iteration. (B) The green task, with conservative processing time, is scheduled to finish at $t'' = 2$ in iteration $\sigma - 2$. In iteration $\sigma - 1$, a delay of 1 h is observed. The processing time delay buffer is maintained, and now the task is anticipated to finish at $t' = 2$. In the next iteration (σ), the task correctly finishes at $t = 0$, accounting for the 1 h delay. The red task can now start on time at $t = 0$, or equivalently $t'' = 2$. Thus, having a delay buffer was useful, since it predicted a realistic start time for the red task in iteration $\sigma - 2$ itself.

We can write the above equations, compactly together, as follows:

$$\bar{W}^n_{ij(t+1)} = \bar{W}^{n-1}_{ijt} - \hat{Y}^{n-1}_{ijt} + \hat{Y}^n_{ijt} \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1, 2, ..., \theta^\tau_{ijt}\} \tag{42}$$

$$\bar{B}^n_{ij(t+1)} = \bar{B}^{n-1}_{ijt} - {}^B\hat{Y}^{n-1}_{ijt} + {}^B\hat{Y}^n_{ijt} \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1, 2, ..., \theta^\tau_{ijt}\} \tag{43}$$

$$\sum_{i \in \mathbf{I}_j} \bar{W}^0_{ijt} + \sum_{i \in \mathbf{I}_j} \sum_{n=0}^{\theta^\tau_{ijt}-1} \bar{W}^n_{ijt} \leq 1 - \hat{\Lambda}_{jt} \quad \forall\, j, t \tag{44}$$

$$S_{k(t+1)} = S_{kt} + \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^+_k} (\bar{\rho}_{ik} \bar{B}^{\theta^\tau_{ijt}}_{ijt} + \hat{\beta}^P_{ijkt}) + \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^-_k} (\rho_{ik} B_{ijt} + \hat{\beta}^C_{ijkt}) - V_{kt} + \zeta_{kt} \quad \forall\, k, t \tag{45}$$

, where

$$\theta^\tau_{ijt} = \begin{cases} \tau_{ij} & \forall j, i \in \mathbf{I}_j, t = \{0\}; \\ \tau^r_{ij} & \forall j, i \in \mathbf{I}_j, t = \{1, 2, 3, ...\} \end{cases} \tag{46}$$

Finally, please note that in this approach, as can be seen in Fig. 9B (iteration $\sigma - 1$), an a priori fixed buffer time ($\tau^r_{ij} - \tau_{ij}$) is added to the task duration, irrespective of whether delays have already been observed (during the execution of the current task). It can be argued that the buffer was meant to absorb the actual delays, and hence, should be cut back for tasks that actually get delayed. This is a fair critique. However, owing to feedback, the true task-finish is accounted for (see Fig. 9A iteration σ), and there is no wasted equipment time due to the unused buffer, which otherwise results in idle time in static robust scheduling approaches.

3.6. Feedback on yield estimates

The material handling loss ($\hat{\beta}^C_{ijkt}/\hat{\beta}^P_{ijkt}$) disturbance parameters can be used to represent yield losses as well, but these parameters are assigned a value only at $t = 0$, i.e., when a task actually ends and a material handling loss is observed. In many applications, the actual yield of a task can, in fact, be estimated during the task's execution, and does not always come as a surprise when the batch finishes. To incorporate the information about anticipated yield loss in future, these parameters can be assigned values for $t > 0$. However, this information has to be then carried over from one iteration to the next, with corresponding decrement in the t index of these parameters to reflect the shifting time-grid, till the task finishes. In addition, if the task is delayed, these parameter values also have to be delayed. This requires cumbersome mechanisms to handle this information in between the online iterations.

Adapting the state-space model to lift the yield-loss information forward provides a much more natural way to handle this feedback. Thus, we define a new free variable $\bar{L}^n_{ijt}$, which is analogous to the batch-size variable $\bar{B}^n_{ijt}$, but, as we can see in Eq. 47, when the task finishes, instead of adding to, it subtracts from the inventory.

$$\begin{aligned} S_{k(t+1)} = S_{kt} &+ \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^+_k} (\bar{\rho}_{ik}(\bar{B}^{\tau_{ij}}_{ijt} - \bar{L}^{\tau_{ij}}_{ijt}) + \hat{\beta}^P_{ijkt}) \\ &+ \sum_j \sum_{i \in \mathbf{I}_j \cap \mathbf{I}^-_k} (\rho_{ik} B_{ijt} + \hat{\beta}^C_{ijkt}) - V_{kt} + \zeta_{kt} \ \ \forall\, k,t \end{aligned} \tag{47}$$

Thus, in a way, this new variable can be thought to be the negative counterpart of the batch-size variable. The yield loss can also be delayed by use of the parameters ${}^L\dot{Y}^n_{ij}$ and ${}^L\hat{Y}^n_{ijt}$ so as to stay in sync with the task-finish time, or nullified in case of a unit breakdown through the parameter ${}^L\dot{Z}^n_{ij}$. This is achieved using the update steps (Eqs. 48-49) and the model lifting equations (Eqs. 50-52).

$$ {}_{\sigma}\bar{L}^n_{ij(t=0)} = {}_{(\sigma-1)}\bar{L}^{n-1}_{ij(t'=0)} + \dot{\lambda}^{n-1}_{ij} - {}^L\dot{Y}^{n-1}_{ij} + {}^L\dot{Y}^n_{ij} - {}^L\dot{Z}^{n-1}_{ij} \quad \forall j, i \in \mathbf{I}_j, n \in \{1, 2, ..., \tau_{ij}\} \tag{48}$$

$$ {}^L_{\sigma}X_{ij(t=0)} = {}^L\dot{Y}^0_{ij} \quad \forall j, i \in \mathbf{I}_j \tag{49}$$

$\dot{\lambda}^n_{ij}$ is a parameter the denotes the "additional" yield loss observed (anticipated) in that iteration (σ). When delays and yield losses are observed simultaneously, then the parameters ${}^L\dot{Y}^n_{ij}$ and ${}^L\dot{Y}^n_{ij}$ take the value ${}_{(\sigma-1)}\bar{L}^{n-1}_{ij(t'=0)} + \dot{\lambda}^{n-1}_{ij}$, i.e., the total yield loss up to and including in iteration σ.

$$ {}^L X_{ij(t+1)} = {}^L\hat{Y}^0_{ijt} \quad \forall j, i \in \mathbf{I}_j, t \tag{50}$$

$$ \bar{L}^0_{ijt} = {}^L X_{ijt} \quad \forall j, i \in \mathbf{I}_j, t \tag{51}$$

$$ \bar{L}^n_{ij(t+1)} = \bar{L}^{n-1}_{ijt} - {}^L\hat{Y}^{n-1}_{ijt} + {}^L\hat{Y}^n_{ijt} \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1, 2, ..., \tau_{ij}\} \tag{52}$$

Please note that there are no L_{ijt} variables in Eq. 51. If these variables were to be present, they would have to be fixed to zero. Otherwise the optimization itself can spuriously assign non-zero values to these variables, such that the intended true batch-size is $B_{ijt} - L_{ijt}$.

3.7. Task Termination

In the update step (Eqs. 20-21, 27-28) we made an implicit assumption that past decisions (task-states with $n > 0$ at $t = 0$) are fixed. In general, more decisions from the previous iteration can be considered fixed or the deviation from them penalized in the current iteration. This has been suggested in the literature to reduce schedule nervousness [23,72,73].

However, to the best knowledge of the authors, no model to date considers canceling/terminating tasks already underway, as an optimization decision. This is surprising, given that whenever needed, this is a very natural decision a human scheduler would take. For example, to prioritize processing for a new rush order, an unfinished process unrelated to this rush order may need to be terminated. Other routine possibilities for task termination are excessive delays or yield losses, as is commonly the case in bio-manufacturing [74,75]. This decision to terminate should be an outcome of the online optimization so as to best react to the observed disturbances or new information. We define task *termination* as a "willful" decision to discontinue a task. This is in contrast with task *suspension*, which is a "forced" discontinuation of a task as a result of a unit breakdown or loss of utility support. Since, preemption is not customary in chemical processes, we assume a total loss of output of a task that is terminated. This is in agreement with how the output of task suspensions is treated.

This termination of tasks can be achieved by "softening" the initial conditions (task-states at $t = 0$) in an online iteration. We introduce a new binary variable, T^n_{ij}, which, when 1, denotes termination of task i, on unit j, which has run-index n. Since, all variables values for iteration $\sigma - 1$ are now a parameter for iteration σ, we can write the following linear equations to achieve this softening:

$$
\begin{aligned}
{}_{\sigma}\bar{W}^{n}_{ij(t=0)} &= \left({}_{(\sigma-1)}\bar{W}^{n-1}_{ij(t'=0)} - \dot{Y}^{n-1}_{ij}\right)(1 - T^{n-1}_{ij}) && (53)\\
&\quad + \dot{Y}^{n}_{ij}(1 - T^{n}_{ij}) - \dot{Z}^{n-1}_{ij} \quad \forall j, i \in \mathbf{I}_j, n \in \{1, 2, ..., \tau_{ij}\} \\
{}_{\sigma}X_{ij(t=0)} &= \dot{Y}^{0}_{ij}(1 - T^{0}_{ij}) \quad \forall j, i \in \mathbf{I}_j && (54)\\
{}_{\sigma}\bar{B}^{n}_{ij(t=0)} &= \left({}_{(\sigma-1)}\bar{B}^{n-1}_{ij(t'=0)} - {}^{B}\dot{Y}^{n-1}_{ij}\right)(1 - T^{n-1}_{ij}) && (55)\\
&\quad + {}^{B}\dot{Y}^{n}_{ij}(1 - T^{n}_{ij}) - {}^{B}\dot{Z}^{n-1}_{ij} \quad \forall j, i \in \mathbf{I}_j, n \in \{1, 2, ..., \tau_{ij}\} \\
{}^{B}_{\sigma}X_{ij(t=0)} &= {}^{B}\dot{Y}^{0}_{ij}(1 - T^{0}_{ij}) \quad \forall j, i \in \mathbf{I}_j && (56)
\end{aligned}
$$

Hence, the update equations (Eqs. 20-21, 27-28 modified to 53-56) now become part of the model. The update equations for inventory (Eq. 18) and backlog (Eq. 19) stay unmodified, and are not softened.

If we had no disturbances, just softening the initial task-states would have sufficed. However, when we have delays, we have to also ensure that we appropriately nullify the effect of delay parameters, which have been already assigned a value at the start of an iteration (optimization). Thus, wherever the delay parameters $\hat{Y}^n_{ijt}$ appear, we multiply these with $(1 - T^n_{ij})$. Since the coefficients of the $(1 - T^n_{ij})$ terms are the delay parameters, when there is no delay, these terms are also, consequently, absent. Parameters $\dot{Y}^{\tau_{ij}}_{ij}$, ${}^{B}\dot{Y}^{\tau_{ij}}_{ij}$, $\hat{Y}^{\tau_{ij}}_{ijt}$, and ${}^{B}\hat{Y}^{\tau_{ij}}_{ijt}$ are always zero, hence, variables $T^{\tau_{ij}}_{ij}$ do not participate in the model. A unit breakdown implicitly disrupts a task, hence, in such a situation, the question of termination does not arise. Overall, the lifting equations are modified to (Eqs. 23 and 30 stay unchanged):

$$
\begin{aligned}
X_{ij(t+1)} &= \hat{Y}^{0}_{ijt}(1 - T^{0}_{ij}) \quad \forall j, i \in \mathbf{I}_j, t && (57)\\
\bar{W}^{0}_{ijt} &= W_{ijt} + X_{ijt} \quad \forall j, i \in \mathbf{I}_j, t && (23)\\
\bar{W}^{n}_{ij(t+1)} &= \bar{W}^{n-1}_{ijt} - \hat{Y}^{n-1}_{ijt}(1 - T^{n-1}_{ij}) + \hat{Y}^{n}_{ijt}(1 - T^{n}_{ij}) \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1, 2, ..., \tau_{ij}\} && (58)\\
{}^{B}X_{ij(t+1)} &= {}^{B}\hat{Y}^{0}_{ijt}(1 - T^{0}_{ij}) \quad \forall j, i \in \mathbf{I}_j, t && (59)\\
\bar{B}^{0}_{ijt} &= B_{ijt} + {}^{B}X_{ijt} \quad \forall j, i \in \mathbf{I}_j, t && (30)\\
\bar{B}^{n}_{ij(t+1)} &= \bar{B}^{n-1}_{ijt} - {}^{B}\hat{Y}^{n-1}_{ijt}(1 - T^{n-1}_{ij}) + {}^{B}\hat{Y}^{n}_{ijt}(1 - T^{n}_{ij}) \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1, 2, ..., \tau_{ij}\} && (60)
\end{aligned}
$$

We do not index variable T^n_{ij} with time (t), because it serves no purpose to terminate a task in future ($t > 0$). If a task is already under execution and has to be terminated in the future for some reason, it is always better to terminate it right away ($t = 0$).

In addition to including a cost associated with task termination, $\sum_j \sum_{i\in\mathbf{I}_j} \sum_{n=0}^{\tau_{ij}-1} \alpha^T_{ij} T^n_{ij}$, in the objective, we can also enforce a pre-specified unit downtime (τ^T_j) following every task termination, by including a summation term in the assignment constraints for that many time-points:

$$\sum_{i\in\mathbf{I}_j} \sum_{n=0}^{\tau_{ij}-1} \bar{W}^n_{ijt} + \sum_{i\in\mathbf{I}_j} \sum_{n=0}^{\tau_{ij}-1} T^n_{ij} \leq 1 - \hat{\Lambda}_{jt} \quad \forall\, j, t \in \{0, 1, 2, ..., \tau^T_j - 1\} \tag{61a}$$

$$\sum_{i\in\mathbf{I}_j} \sum_{n=0}^{\tau_{ij}-1} \bar{W}^n_{ijt} \leq 1 - \hat{\Lambda}_{jt} \quad \forall\, j, t \in \{\tau^T_j, \tau^T_j + 1, ...\} \tag{61b}$$

We can write the above two equations, compactly together, as follows:

$$\sum_{i\in\mathbf{I}_j} \sum_{n=0}^{\tau_{ij}-1} \bar{W}^n_{ijt} + \sum_{i\in\mathbf{I}_j} \sum_{n=0}^{\tau_{ij}-1} \theta^T_{ijt} T^n_{ij} \leq 1 - \hat{\Lambda}_{jt} \quad \forall\, j, t \tag{62}$$

, where

$$\theta^T_{ijt} = \begin{cases} 1 \;\; \forall t = \{0, 1, 2, ..., \tau^T_{ij} - 1\}; \\ 0 \;\; \forall t = \{\tau^T_{ij}, \tau^T_{ij} + 1, \tau^T_{ij} + 2, ...\} \end{cases} \tag{63}$$

An added advantage of the compact form, is that the unit downtime can now be a function of the task that was terminated, i.e., τ^T_{ij} is indexed by i as well, and not just j.

To systematically account for unit downtime resulting from task-termination in a previous iteration, i.e., if ${}_{(\sigma-1)}T^n_{ij} = 1 \;\; \forall j, i \in \mathbf{I}_j, n \in \{1, 2, 3, ..., \tau_{ij} - 1\}$., the parameter $\hat{\Lambda}_{jt}$ has to be activated for $t \in \{0, 1, 2, ..., \tau^T_j - 2\}$ in iteration σ. Thereafter, in each subsequent iterations, the downtime is decremented by 1, and the parameter $\hat{\Lambda}_{jt}$ appropriately activated for the corresponding time-points. Alternatively, we can define a new binary variable and lift it, to keep the unit deactivated for the remaining downtime in subsequent iterations. The new variable, which would be subtracted on the right hand side of Eqs. 61a-61b, in lieu of $\hat{\Lambda}_{jt}$, can be thought of as the unit unavailability variable.

3.8. Post-production storage in unit

Kondili et al. (1993) [28] proposed a formulation for "hold" tasks, which are dummy tasks that can be used to model storage of materials in a processing unit, while waiting to be unloaded. This is especially important for production facilities that follow a no intermediate storage (NIS) policy for certain materials. In the network shown in Fig. 1, task T4 is a hold task. The purpose of this task is to keep material M3 residing in unit U3. So as to ensure that the hold task can only store material in a unit that it was originally produced in, we write the state-space version of the original constraint proposed by Kondili et al. (1993) [28] in Eq. 64.

$$B_{i',jt} \leq \bar{B}^{\tau_{i',j}}_{i',jt} + \sum_{i\in\mathbf{I}_j} \bar{B}^{\tau_{ij}}_{ijt} \quad \forall i' \in \mathbf{I}_{\text{HOLD}}, j \in \mathbf{J}_{i'}, t \tag{64}$$

, where $\mathbf{I}_{\text{HOLD}}$ is the set of hold tasks. When multiple materials are produced in a unit, here we assume that the corresponding hold task emulates the simultaneous storage of all these materials in the unit. The $\bar{\rho}_{\text{HOLD},k}$ mass-coefficient then dictates in what proportion are the materials released from the unit. If only a certain individual material is held, and the others unloaded, the term $\sum_{i\in\mathbf{I}_j} \bar{\rho}_{ik} \bar{B}^{\tau_{ij}}_{ijt}$ is substituted with $\sum_{i\in\mathbf{I}^+_k \cap \mathbf{I}_j} \bar{\rho}_{ik} \bar{B}^{\tau_{ij}}_{ijt}$, and the constraint is written only for that material k which is held.

Further, for a perishable material, $\bar{\rho}_{\text{HOLD},k} < \rho_{\text{HOLD},k}$; that is, a fraction of the material is lost (perishes or deactivates) when held.

3.9. Unit capacity degradation and maintenance

In many processes, such as polymerization reactions, or purification processes, it is not uncommon for the unit capacity to "degrade" after a task has been processed on that unit [76–81]. This can be, for example, due to residue formation (e.g. scaling) or impurity accumulation (e.g. membrane pore blockages). Some degradation is gradual and predictable, while some degradation may occur suddenly and unexpectedly.

To model unit degradation, we define a new non-negative variable, C_{ijt}, which denotes the capacity of the unit j, to perform task i that starts at time t. For an un-degraded unit, C_{ijt} is initialized to the value β_{ij}^{max}. This variable value is passed over from one iteration to another, through the update equation:

$$ {}_{\sigma}C_{ij(t=0)} = {}_{(\sigma-1)}C_{ij(t'=1)} + \dot{\mu}_{ij} \quad \forall j \in \mathbf{J}_{\text{MT}}, i \in \mathbf{I}_j \tag{65}$$

A new disturbance parameter, $\dot{\mu}_{ij}$, which when negative, represents extent of sudden (unexpected) partial loss in unit capacity. Conversely, a positive value represents renewal of unit capacity. This positive value could be, for example, a result of installing a new repaired unit in place of an old unit that broke down.

Through Eq. 66, we define a balance on the unit capacity. $\rho_{ii'j}^{C}$ is a parameter that denotes the gradual degradation in capacity of unit j to perform task i, due to execution of task i' on that unit. $\rho_{ii'j}^{C}$ is either negative or zero.

$$ C_{ij(t+1)} = C_{ijt} + \sum_{i' \in \mathbf{I}_j} \sum_{n=1}^{\tau_{i'j}} \frac{\rho_{ii'j}^{C}}{\tau_{i'j}} \bar{B}_{i'jt}^{n} + \bar{M}_{ijt}^{\tau_{\text{MT},j}} \quad \forall j \in \mathbf{J}_{\text{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\text{MT}}\}, t \tag{66}$$

The degraded unit can be typically restored to its full capacity through a maintenance task (e.g. cleaning), which we denote with the abbreviation MT. We define $\mathbf{I}_{\text{MT}}$ as a set of maintenance tasks, and $\mathbf{J}_{\text{MT}}$ as a set of units which can degrade, and consequently, need a corresponding maintenance task. Further, we add the maintenance task (MT) to the set of tasks $\mathbf{I}_j$ that can be performed on unit j. The value of $\bar{M}_{ijt}^{\tau_{\text{MT},j}}$, the variable which we define below, dictates the restored capacity due to completion of a maintenance task.

Like any conventional task, the maintenance task has a start binary ($W_{\text{MT},jt}$) associated with it, which is appropriately lifted. The assignment equation (Eq. 25) ensures that the maintenance task can only run when the unit is not running any other task. Since the maintenance task does not consume or produce materials, the conventional batch-size of this maintenance task, $B_{\text{MT},jt}$ is fixed to zero. Instead, we define a new type of batch-size, $\bar{M}_{ijt}^{n}$, specific to the purpose of this task, which we term as the maintenance-size. Since we assume that only one kind of a maintenance task exists for every unit, the index i in this maintenance-size variable is not MT. This variable denotes, how much capacity to perform task i is restored, when maintenance is performed on unit j. This variable is also lifted, similar to the batch-size variable, and can be delayed or suspended due to breakdowns. Similarly, the parameters ${}^{M}\dot{Y}_{ij}^{n}$, ${}^{M}\hat{Y}_{ijt}^{n}$, and ${}^{M}\dot{Z}_{ij}^{n}$ denote the maintenance-sizes of the maintenance task corresponding to capacity restored to perform task i. The update equations associated with this variable are:

$$ {}_{\sigma}\bar{M}_{ij(t=0)}^{n} = {}_{(\sigma-1)}\bar{M}_{ij(t'=0)}^{n-1} - {}^{M}\dot{Y}_{ij}^{n-1} + {}^{M}\dot{Y}_{ij}^{n} - {}^{M}\dot{Z}_{ij}^{n-1} \quad \forall j \in \mathbf{J}_{\text{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\text{MT}}\}, n \in \{1, 2, ..., \tau_{\text{MT},j}\} \tag{67}$$

$$ {}_{\sigma}^{M}X_{ij(t=0)} = {}^{M}\dot{Y}_{ij}^{0} \quad \forall j \in \mathbf{J}_{\text{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\text{MT}}\} \tag{68}$$

The model equations, for lifting, are:

$$^{M}X_{ij(t+1)} = {}^{M}\hat{Y}^{0}_{ijt} \quad \forall j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t \tag{69}$$

$$\bar{M}^{0}_{ijt} = M_{ijt} + {}^{M}X_{ijt} \quad \forall j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t \tag{70}$$

$$\bar{M}^{n}_{ij(t+1)} = \bar{M}^{n-1}_{ijt} - {}^{M}\hat{Y}^{n-1}_{ijt} + {}^{M}\hat{Y}^{n}_{ijt} \quad \forall j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t, n \in \{1, 2, ..., \tau_{\mathrm{MT},j}\} \tag{71}$$

The batch-size of new tasks is upper bounded by the unit capacity variable (Eq. 72). This ensures that only smaller batches can now be processed if $C_{ijt} < \beta^{max}_{ij}$. If a task just finishes, the restored or degraded capacity due to that is also accounted for while upper bounding the task-batchsize B_{ijt}.

$$B_{ijt} \leq C_{ijt} + \sum_{i' \in \mathbf{I}_j} \frac{\rho^{C}_{ii'j}}{\tau_{i'j}} \bar{B}^{\tau_{i'j}}_{i'jt} + \bar{M}^{\tau_{\mathrm{MT},j}}_{ijt} \quad \forall\, j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t \tag{72}$$

We define the maintenance task with fixed processing time ($\tau_{\mathrm{MT},j}$), and assume that whenever performed, restores the unit to its full capacity (i.e. $C_{ijt} = \beta^{max}_{ij}$). This requires the maintenance-size, M_{ijt}, to be the difference between the deteriorated unit capacity (C_{ijt}), before the maintenance starts, and the upper capacity limit (β^{max}_{ij}), to which the unit has to be restored to. This is achieved using Eq. 73:

$$\beta^{max}_{ij} W_{\mathrm{MT},jt} - C_{ijt} \leq M_{ijt} \leq \beta^{max}_{ij} W_{\mathrm{MT},jt} \quad \forall\, j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t \tag{73}$$

Finally, we specify the lower and upper bounds for variable C_{ijt}:

$$0 \leq C_{ijt} \leq \beta^{max}_{ij} \quad \forall\, j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t \tag{74}$$

, and define a new parameter α^{M}_{ij}, which denotes the proportional cost of maintenance of a unit. The cost term, in the objective, for a maintenance task is $\sum_{j \in \mathbf{J}_{\mathrm{MT}}} \sum_{t} (\alpha^{F}_{\mathrm{MT},j} W_{\mathrm{MT},jt} + \sum_{i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}} \alpha^{M}_{ij} M_{ijt})$.

4. Integrated model

In this section, we present the complete model with all generalizations present simultaneously. For brevity, we write index σ in only those equations, where $\sigma - 1$ is also present. Everywhere else, all variables are those of the current iteration σ.

The update equations are:

$${}_{\sigma}S_{k(t=0)} = {}_{(\sigma-1)}S_{k(t'=1)} \quad \forall k \tag{18}$$

$${}_{\sigma}BO_{k(t=0)} = {}_{(\sigma-1)}BO_{k(t'=1)} \quad \forall k \tag{19}$$

$${}_{\sigma}C_{ij(t=0)} = {}_{(\sigma-1)}C_{ij(t'=1)} + \dot{\mu}_{ij} \quad \forall j \in \mathbf{J}_{\mathrm{MT}}, i \in \mathbf{I}_j \tag{65}$$

The softened update equations, which are part of the model, are:

$$\begin{aligned}
{}_\sigma\bar{W}^n_{ij(t=0)} &= \left({}_{(\sigma-1)}\bar{W}^{n-1}_{ij(t'=0)} - \dot{Y}^{n-1}_{ij}\right)(1-T^{n-1}_{ij}) \\
&\quad + \dot{Y}^n_{ij}(1-T^n_{ij}) - \dot{Z}^{n-1}_{ij} \quad \forall j, i \in \mathbf{I}_j, n \in \{1,2,...,\tau_{ij}\} && (53)\\
{}_\sigma X_{ij(t=0)} &= \dot{Y}^0_{ij}(1-T^0_{ij}) \quad \forall j, i \in \mathbf{I}_j && (54)\\
{}_\sigma\bar{B}^n_{ij(t=0)} &= \left({}_{(\sigma-1)}\bar{B}^{n-1}_{ij(t'=0)} - {}^B\dot{Y}^{n-1}_{ij}\right)(1-T^{n-1}_{ij}) \\
&\quad + {}^B\dot{Y}^n_{ij}(1-T^n_{ij}) - {}^B\dot{Z}^{n-1}_{ij} \quad \forall j, i \in \mathbf{I}_j, n \in \{1,2,...,\tau_{ij}\} && (55)\\
{}^B_\sigma X_{ij(t=0)} &= {}^B\dot{Y}^0_{ij}(1-T^0_{ij}) \quad \forall j, i \in \mathbf{I}_j && (56)\\
{}_\sigma\bar{L}^n_{ij(t=0)} &= \left({}_{(\sigma-1)}\bar{L}^{n-1}_{ij(t'=0)} + \dot{\lambda}^{n-1}_{ij} - {}^L\dot{Y}^{n-1}_{ij}\right)(1-T^{n-1}_{ij}) \\
&\quad + {}^L\dot{Y}^n_{ij}(1-T^n_{ij}) - {}^L\dot{Z}^{n-1}_{ij} \quad \forall j, i \in \mathbf{I}_j, n \in \{1,2,...,\tau_{ij}\} && (75)\\
{}^L_\sigma X_{ij(t=0)} &= {}^L\dot{Y}^0_{ij}(1-T^0_{ij}) \quad \forall j, i \in \mathbf{I}_j && (76)\\
{}_\sigma\bar{M}^n_{ij(t=0)} &= {}_{(\sigma-1)}\bar{M}^{n-1}_{ij(t'=0)} - {}^M\dot{Y}^{n-1}_{ij}(1-T^{n-1}_{\mathrm{MT},j}) \\
&\quad + {}^M\dot{Y}^n_{ij}(1-T^n_{\mathrm{MT},j}) - {}^M\dot{Z}^{n-1}_{ij} \quad \forall j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, n \in \{1,2,...,\tau_{\mathrm{MT},j}\} && (77)\\
{}^M_\sigma X_{ij(t=0)} &= {}^M\dot{Y}^0_{ij}(1-T^0_{\mathrm{MT},j}) \quad \forall j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\} && (78)
\end{aligned}$$

The lifting equations are:

$$\begin{aligned}
X_{ij(t+1)} &= \hat{Y}^0_{ijt}(1-T^0_{ij}) \quad \forall j, i \in \mathbf{I}_j, t && (57)\\
\bar{W}^0_{ijt} &= W_{ijt} + X_{ijt} \quad \forall j, i \in \mathbf{I}_j, t && (23)\\
\bar{W}^n_{ij(t+1)} &= \bar{W}^{n-1}_{ijt} - \hat{Y}^{n-1}_{ijt}(1-T^{n-1}_{ij}) + \hat{Y}^n_{ijt}(1-T^n_{ij}) \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1,2,...,\theta^\tau_{ijt}\} && (79)\\
{}^B X_{ij(t+1)} &= {}^B\hat{Y}^0_{ijt}(1-T^0_{ij}) \quad \forall j, i \in \mathbf{I}_j, t && (59)\\
\bar{B}^0_{ijt} &= B_{ijt} + {}^B X_{ijt} \quad \forall j, i \in \mathbf{I}_j, t && (30)\\
\bar{B}^n_{ij(t+1)} &= \bar{B}^{n-1}_{ijt} - {}^B\hat{Y}^{n-1}_{ijt}(1-T^{n-1}_{ij}) + {}^B\hat{Y}^n_{ijt}(1-T^n_{ij}) \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1,2,...,\theta^\tau_{ijt}\} && (80)\\
{}^L X_{ij(t+1)} &= {}^L\hat{Y}^0_{ijt}(1-T^0_{ij}) \quad \forall j, i \in \mathbf{I}_j, t && (81)\\
\bar{L}^0_{ijt} &= {}^L X_{ijt} \quad \forall j, i \in \mathbf{I}_j, t && (51)\\
\bar{L}^n_{ij(t+1)} &= \bar{L}^{n-1}_{ijt} - {}^L\hat{Y}^{n-1}_{ijt}(1-T^{n-1}_{ij}) + {}^L\hat{Y}^n_{ijt}(1-T^n_{ij}) \quad \forall j, i \in \mathbf{I}_j, t, n \in \{1,2,...,\theta^\tau_{ijt}\} && (82)\\
{}^M X_{ij(t+1)} &= {}^M\hat{Y}^0_{ijt}(1-T^0_{\mathrm{MT},j}) \quad \forall j \in \mathbf{J}_{\mathrm{MT}}, i \in \mathbf{I}_j, t && (83)\\
\bar{M}^0_{ijt} &= M_{ijt} + {}^M X_{ijt} \quad \forall j \in \mathbf{J}_{\mathrm{MT}}, i \in \mathbf{I}_j, t && (70)\\
\bar{M}^n_{ij(t+1)} &= \bar{M}^{n-1}_{ijt} - {}^M\hat{Y}^{n-1}_{ijt}(1-T^{n-1}_{\mathrm{MT},j}) \\
&\quad + {}^M\hat{Y}^n_{ijt}(1-T^n_{\mathrm{MT},j}) \quad \forall j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t, n \in \{1,2,...,\theta^\tau_{\mathrm{MT},jt}\} && (84)
\end{aligned}$$

424 , where please note the use of parameter θ^τ_{ijt} in Eqs. 79, 80, 82, and 84.

The assignment constraint is:

$$\sum_{i \in \mathbf{I}_j} \sum_{n=0}^{\theta^\tau_{ijt}-1} W^n_{ijt} + \sum_{i \in \mathbf{I}_j} \sum_{n=0}^{\theta^\tau_{ijt}-1} \theta^T_{ijt} T^n_{ij} \leq 1 - \hat{\Lambda}_{jt} \quad \forall\, j, t \tag{85}$$

The inventory, backlog, and unit capacity balance are:

$$S_{k(t+1)} = S_{kt} + \sum_{j} \sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^+} (\theta^{\bar{\rho}}_{ikt} (\bar{B}_{ijt}^{\theta^\tau_{ijt}} - \bar{L}_{ijt}^{\theta^\tau_{ijt}}) + \hat{\beta}^P_{ijkt}) \tag{86}$$

$$+ \sum_{j} \sum_{i \in \mathbf{I}_j \cap \mathbf{I}_k^-} (\rho_{ik} B_{ijt} + \hat{\beta}^C_{ijkt}) - V_{kt} + \zeta_{kt} \quad \forall\, k, t$$

$$BO_{k(t+1)} = BO_{kt} - V_{kt} + \hat{\xi}_{kt} \quad \forall\, k, t \tag{15}$$

$$C_{ij(t+1)} = C_{ijt} + \sum_{i' \in \mathbf{I}_j} \sum_{n=1}^{\theta^\tau_{i'jt}} \frac{\rho^C_{ii'j}}{\theta^\tau_{i'jt}} \bar{B}^n_{i'jt} + \bar{M}_{ijt}^{\theta^\tau_{\mathrm{MT},jt}} \quad \forall\, j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t \tag{87}$$

Batch-size, post-production storage, and maintenance-size constraints are:

$$\beta^{min}_{ij} W_{ijt} \le B_{ijt} \le \beta^{max}_{ij} W_{ijt} \quad \forall\, j, i \in \mathbf{I}_j, t \tag{2}$$

$$B_{i',jt} \le \bar{B}_{i',jt}^{\theta^\tau_{i',jt}} + \sum_{i \in \mathbf{I}_j} \bar{B}_{ijt}^{\theta^\tau_{ijt}} \quad \forall i' \in \mathbf{I}_{\mathrm{HOLD}}, j \in \mathbf{J}_{i'}, t \tag{88}$$

$$B_{ijt} \le C_{ijt} + \sum_{i' \in \mathbf{I}_j} \frac{\rho^C_{ii'j}}{\theta^\tau_{i'j}} \bar{B}_{i'jt}^{\theta^\tau_{i'j}} + \bar{M}_{ijt}^{\theta^\tau_{\mathrm{MT},j}} \quad \forall\, j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t \tag{89}$$

$$\beta^{max}_{ij} W_{\mathrm{MT},jt} - C_{ijt} \le M_{ijt} \le \beta^{max}_{ij} W_{\mathrm{MT},jt} \quad \forall\, j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t \tag{73}$$

Bounds on the variables are as follows:

$$0 \le C_{ijt} \le \beta^{max}_{ij} \quad \forall\, j \in \mathbf{J}_{\mathrm{MT}}, i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}, t \tag{74}$$

$$W_{ijt}, \bar{W}^n_{ijt}, T^n_{ij} \in \{0,1\};\; B_{ijt}, M_{ijt}, \bar{B}^n_{ijt}, S_{kt}, BO_{kt}, V_{kt} \ge 0; \tag{90}$$

$$\bar{W}^n_{ij(t=1)}, \bar{B}^n_{ij(t=1)}, \bar{L}^n_{ij(t=1)} = 0\; \forall n = \{\tau_{ij}+1, \tau_{ij}+2, ..., \tau^r_{ij}\} \tag{91}$$

$$\bar{M}^n_{ij(t=1)} = 0\; \forall n = \{\tau_{\mathrm{MT},j}+1, \tau_{\mathrm{MT},j}+2, ..., \tau^r_{\mathrm{MT},j}\} \tag{92}$$

Variables $X_{ijt}, {}^{B}X_{ijt}, {}^{L}X_{ijt}, {}^{M}X_{ijt}$ are free variables, however, through the update and model equations, they are always equated to a parameter value, hence, are not degrees of freedom.

The objective is:

$$\min \sum_k \sum_t (\gamma^{INV}_k S_{kt} + \gamma^{BO}_k BO_{kt}) + \sum_j \sum_{i \in \mathbf{I}_j} \sum_t (\alpha^F_{ij} W_{ijt} + \alpha^P_{ij} B_{ijt}) \tag{93}$$

$$+ \sum_j \sum_{i \in \mathbf{I}_j} \sum_{n=0}^{\tau_{ij}-1} \alpha^T_{ij} T^n_{ij} + \sum_{j \in \mathbf{J}_{\mathrm{MT}}} \sum_{i \in \{\mathbf{I}_j \setminus \mathbf{I}_{\mathrm{MT}}\}} \sum_t \alpha^M_{ij} M_{ijt}$$

5. Case study

In this section, we present an example, to demonstrate all modeling generalizations discussed in Section 3, using the integrated model equations outlined in Section 4. We employ the network of Fig. 1. In general, bio-manufacturing processes can be divided into an upstream bio-reaction (e.g. fermentation) stage and a downstream purification stage [74]. The upstream stage typically consists of two steps: cell culture preparation in the lab and the bio-reaction. These two steps are task T1 and T2, respectively, in the network in Fig. 1. The downstream purification stage typically consists of three steps: centrifugation, chromatography, and filtration. Among these three steps, chromatography takes the longest and the chromatograph columns are prone to unpredictable failures. Hence, we assume that chromatography, being the dominant step, is representative of the complete purification stage.

This is task T3 in the network in Fig. 1. Overall, we choose this simplified system (network) for an easier illustration of the different features of our general state-space model.

In bio-manufacturing, the molecules of interest, rather than being chemically synthesized, are produced through living organisms. The use of live systems such as bacteria, mammalian, or insect cells in the upstream stages, introduces several operational challenges. These include batch-to-batch variability, parallel growth of both, the desired antibodies as well as the unwanted toxic byproducts in the same batch, and possible random shocks that can lead to complete failure of a batch [75]. In our example, as features, we allow for possible delays in the cell culture preparation (T1), and small yield losses in the bio-reaction (T2), including possible substantial yield losses due to sudden cell death. Thus, we carry out robust scheduling, using a conservative processing time (τ^r) for task T1 and a conservative mass-conversion coefficient ($\bar{\rho}^r$), against small yield losses, for task T2. Further, in the downstream stage, we assume that the chromatograph column (U3) loses capacity with usage, part of which is predictable. Executing a maintenance task can restore capacity on the chromatographs. To enforce the no-intermediate storage (NIS) policy for material M2, the inventory variable $S_{\text{M2},t}$ is fixed to zero for all time-points. Raw material, M0, is assumed to be available in an unrestricted supply, as needed. Select instance parameter values, other than the ones already shown in Fig. 1, are outlined in Table 1.

Table 1. Parameter values for the case study.

$\gamma_k^{INV} = 0.15\gamma_k$	$\gamma_k^{BO} = 1.5\gamma_k$	$\alpha_{ij}^F = 1$	$\alpha_{ij}^P = 0$	$\alpha_{ij}^T = 2$	$\alpha_{ij}^M = 0$
$\tau_{\text{U3}}^T = 2$	$\tau_{\text{T1,U1}}^r = 3$	$\tau_{\text{MT,U3}} = 2$	$\bar{\rho}_{\text{M2,T2}}^r = 0.9$	$\rho_{\text{T3,T3,U3}}^C = -0.2$	$\delta = 1$

An order for 15 kg of M3 is due at the 14th hour of the day. To meet this order, online scheduling is carried out, with a horizon of 16 h, and re-optimization every 1 h, starting at the 0th hour. All optimizations are solved to optimality using default solver options in CPLEX 12.6.1 via GAMS 24.4.3, installed on an Intel Xeon (E5520, 2.27GHz, 8 core processor) machine, with 16 GB of RAM and Linux CentOS 7 operating system. The schedules obtained in select online scheduling iteration are shown in Fig. 10. For the remaining online iterations, the predicted schedule is identical to the respective previous iteration (but with time-grid shift).

The nominal makespan, without any disturbances or robustification, to meet this order is 13 h. But, in iteration 0 (see Fig. 10), T1 is started at $t = 0$, instead of $t = 1$, since a conservative processing time, $\tau^r = 3$ h is in use. Further, the batch-sizes for T1 and T2 are 16.67 kg, since a conservative yield parameter ($\bar{\rho}^r = 0.9$) is in use for T2. This predicts production of 15 kg of M2, which through the two T3 batches, of 6.25 kg and 8.75 kg, makes 15 kg of M3.

In the next iteration, a fractional delay of 0.5 h is observed. Due to the use of a discrete time-grid with granularity δ =1 h, and being the first delay for this task, this is rounded up to 1 h. Since, now the order is predicted to be late, the batch-sizes of T3 are revised so as to meet as much of the order as possible on time ($\beta^{max} = 10$). Going forward, no more delays are observed in T1, hence, it finishes at $t = 0$ in iteration 3. This is because the nominal τ is in use at $t = 0$ in Eqs. 85 and 86. Consequently, the downstream tasks are all scheduled earlier now, matching up with the initial predicted schedule in iteration 0. Thus, the conservative processing time was useful, in making T1 start earlier at 0th hour.

In iteration 4, due to sudden cell death, 90% yield loss in T2 is observed (anticipated at task finish). Hence, T2 is terminated. T1 is restarted (with conservative processing time). Since, no delays are observed through the execution of this new task T1, in iteration 6, it finishes after the nominal processing time of 2 h. Thus the start times of the downstream tasks are pulled forward by 1 h.

In iteration 9, 10% yield loss is observed (anticipated at task finish). Since, this is not substantial, T2 is not terminated. Instead, a new train of tasks is scheduled to start at $t = 2$ to compensate for the lost yield. In iteration 11, when T2 finishes, it results in nominal yield ($\bar{\rho}$) minus the 10% anticipated yield loss. Hence, 15 kg of M2 is produced. Thus, the new train of tasks previously scheduled, but not yet started, in iteration 9 are canceled.

In iteration 11, in addition to the gradual decline in chromatograph capacity due to usage ($\rho^{C}_{\mathrm{T3,T3,U3}} < 0$), which does not affect starting the next 5 kg T3 task, a sudden loss in capacity is observed ($\dot{\mu}_{\mathrm{T3,U3}} = -8$). Hence, a maintenance task (MT) is scheduled with maintenance-size $M_{\mathrm{T3,T3},t=1} = 9.34$. Once this maintenance is over, the pending task T3 can start. No further disturbances are observed. Consequently, the order is fully met at the 19th hour.

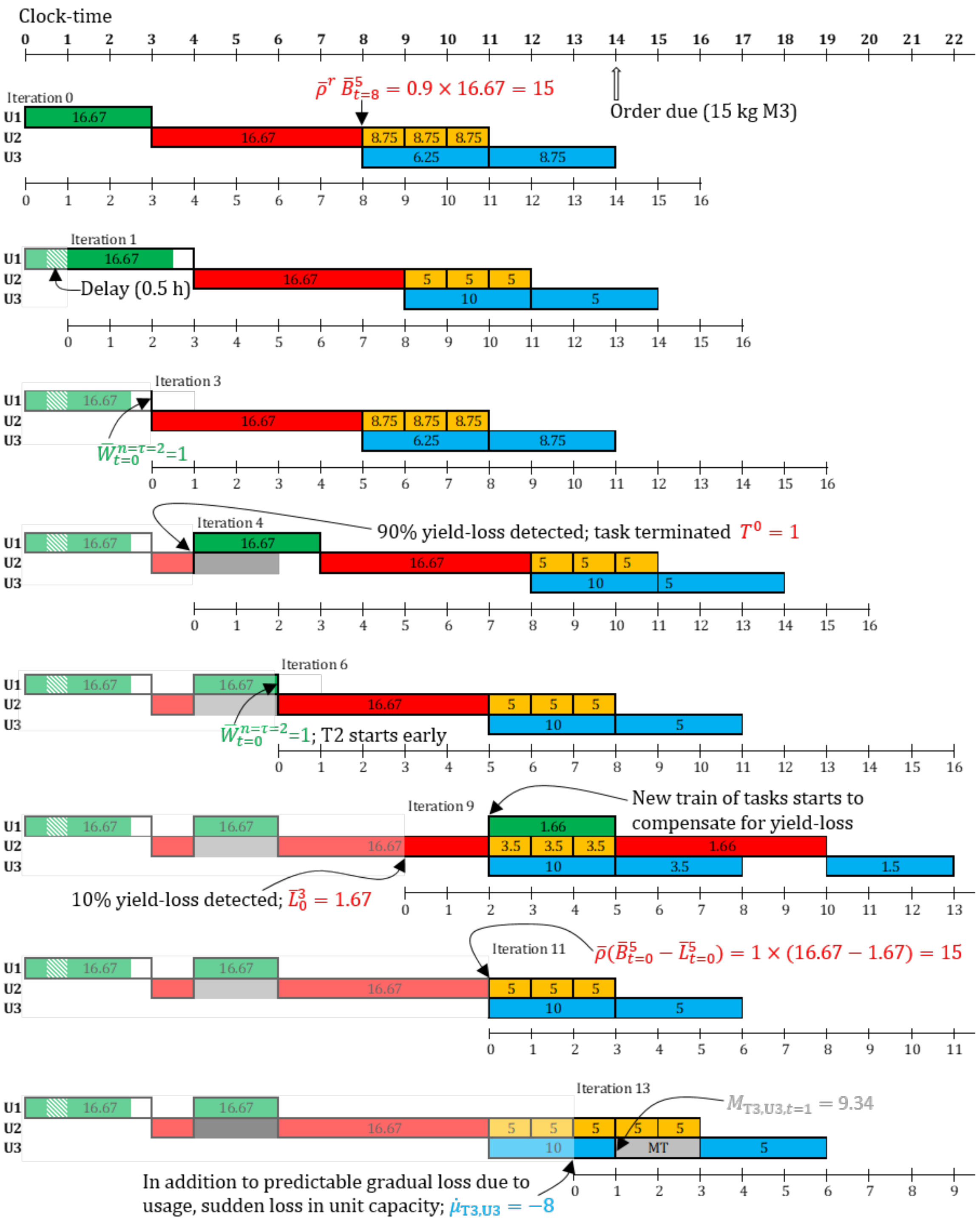


Figure 10. Select online scheduling iterations for the case study. The color of the tasks corresponds to their color in Fig. 1. Executed schedule, with respect to each iteration, is shown in lighter (fainted) colors. The clock-time is the global time. Each iteration has its own local discrete time-grid.

6. Closing remarks

We developed a general state-space model, particularly motivated by an online scheduling perspective, that allows modeling (1) task-delays and unit breakdowns with a new, more intuitive convention over that of Subramanian et al. (2012) [24], (2) fractional delays and unit downtimes, when using discrete-time grid, (3) variable batch-sizes, (4) robust scheduling through the use of conservative yield estimates and processing times, (5) feedback on task-yield estimates before the task finishes, (6) task termination during its execution, (7) post-production storage of material in unit, and (8) unit capacity degradation and maintenance. Further, we propose a new scheme for updating the state of the process, as well as an overall formulation to enforce constraints (through parameter/variable modifications), based on feedback information, on future decisions. We demonstrate the effectiveness of this model on a case study from the field of bio-manufacturing. Through this new state-space model, we have enabled a natural way to handle routinely encountered processing features and disturbance information in online scheduling, in general. The proposed model, therefore, greatly extends and enables the possible application of mathematical programming based online scheduling solutions to diverse application settings, including bio-manufacturing. Finally, it is important to note, that although here we presented the model using STN based representation, these generalizations can also be adapted to RTN based representation.

Acknowledgments: The authors would like to acknowledge support from the National Science Foundation under grants CMMI-1334933 and CBET-1264096, as well as the Petroleum Research Fund under grant 53313-ND9.

Author Contributions: D.G. conceived the model and prepared the manuscript under the supervision of C.T.M.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MILP	Mixed integer linear program
RTN	Resource task network
STN	State task network
HOLD	Hold (storage) task
MT	Maintenance (cleaning) task

Nomenclature

Indices/sets

$i \in \mathbf{I}$	tasks
$j \in \mathbf{J}$	units (equipment)
$k \in \mathbf{K}$	materials
$t \in \mathbf{T}$	time-points/periods
$\mathbf{I} \supseteq \mathbf{I}_j$	tasks that can be carried out in unit j
$\mathbf{I} \supseteq \mathbf{I}_k^+$	tasks producing material k
$\mathbf{I} \supseteq \mathbf{I}_k^-$	tasks consuming material k
$\mathbf{I} \supseteq \mathbf{I}_{\text{HOLD}}$	hold (storage) tasks
$\mathbf{I} \supseteq \mathbf{I}_{\text{MT}}$	maintenance tasks
$\mathbf{J} \supseteq \mathbf{J}_i$	units suitable for carrying out task i
$\mathbf{J} \supseteq \mathbf{J}_{\text{MT}}$	units which can degrade, and consequently, need a corresponding maintenance task
$\mathbf{K} \supseteq \mathbf{K}^F$	feed (raw) materials
$\mathbf{K} \supseteq \mathbf{K}^I$	intermediates
$\mathbf{K} \supseteq \mathbf{K}^P$	final products

Parameters

α_{ij}^F	fixed cost of running task i on unit j

α^P_{ij}	proportional cost of running task i on unit j
α^T_{ij}	cost of terminating task i on unit j
α^M_{ij}	proportional cost of maintenance task i on unit j
β_{ij}	fixed batch-size of task i executed on unit j
β^{min}_{ij} / β^{max}_{ij}	min/max capacity on batch-size of task i executed on unit j
$\hat{\beta}^P_{ijkt}$ / $\hat{\beta}^C_{ijkt}$	material unloading/loading loss during production/consumption of material k
γ_k	selling price of material k
γ^{INV}_k	inventory cost of material k
γ^{BO}_k	backlog cost of material k
δ	discretization of time-grid; length of time-periods
ζ_{kt}	incoming shipment of material k at time t
$\dot{Z}^n_{ij}$	disturbance parameter denoting unit breakdown
${}^B\dot{Z}^n_{ij}$	batch-size of task suspended due to unit breakdown
${}^L\dot{Z}^n_{ij}$	yield-loss size of task suspended due to unit breakdown
${}^M\dot{Z}^n_{ij}$	maintenance-size of maintenance task suspended due to unit breakdown
θ^{τ}_{ijt}	dummy parameter as defined in Eq. 46
θ^{ρ}_{ikt}	dummy parameter as defined in Eq. 37
θ^{T}_{ijt}	dummy parameter as defined in Eq. 63
$\dot{\lambda}^n_{ij}$	yield loss in task i running on unit j, with run status n
$\hat{\Lambda}_{jt}$	binary parameter, when 1, denotes unit j unavailable during time $[t, t+1)$
$\dot{\mu}_{ij}$	when negative, represents extent of sudden partial loss in unit capacity
ξ_{kt}	demand for material k at time t
$\hat{\xi}_{kt}$	demand disturbance for material k at time t
π_{break}	actual (fractional) time at which a unit breaks down
π_{down}	fractional downtime in a unit
π_{delay}	fractional delay in a task
π^r_{delay}	r^{th} delay in a task
ρ_{ik}	mass-conversion coefficient (material consumption)
$\bar{\rho}_{ik}$	mass-conversion coefficient (material production)
$\bar{\rho}^r_{ik}$	conservative mass-conversion coefficient for production ($\bar{\rho}^r_{ik} < \bar{\rho}_{ik}$)
$\rho^C_{ii'j}$	deterioration in unit capacity to perform task i, due to performing task i' on that unit j.
σ	online iteration number
τ_{ij}	processing time of task i on unit j
τ^r_{ij}	conservative processing time of task i ($\tau^r_{ij} < \tau_{ij}$) on unit j
τ^T_j	task independent unit j downtime after terminating a task
τ^T_{ij}	task dependent unit j downtime after terminating task i
$\dot{\Upsilon}^n_{ij}, \hat{\Upsilon}^n_{ijt}$	single-/multi-period disturbance parameters denoting delay
${}^B\dot{\Upsilon}^n_{ij}, {}^B\hat{\Upsilon}^n_{ijt}$	single-/multi-period disturbance parameters denoting batch-size of a delayed task
${}^L\dot{\Upsilon}^n_{ij}, {}^L\hat{\Upsilon}^n_{ijt}$	single-/multi-period disturbance parameters denoting yield-loss size of a delayed task
${}^M\dot{\Upsilon}^n_{ij}, {}^M\hat{\Upsilon}^n_{ijt}$	single-/multi-period disturbance parameters denoting maintenance-size of delayed maintenance task
ϕ	duration of delay or breakdown, in multiples of δ
ψ	recurrence count of delay for a task

Variables

B_{ijt}	batch-size of task i on unit j
$\bar{B}^n_{ijt}$	lifted batch-size
BO_{kt}	backlog level of material k during period $(t-1, t]$
C_{ijt}	capacity of unit j to perform task i during period $(t-1, t]$

$\bar{L}^n_{ijt}$	lifted yield-loss variables
M_{ijt}	maintenance-size of the maintenance task
$\bar{M}^n_{ijt}$	lifted maintenance-size
S_{kt}	inventory level of material k during period $(t-1, t]$
T^n_{ij}	binary variable, when 1, denotes termination of task i, with run-status n, on unit j
V_{kt}	outgoing shipment to meet demand for material k at time t
W_{ijt}	binary variable, when 1, denotes task i starts on unit j at time-point t
$\bar{W}^n_{ijt}$	lifted task-start variables
X_{ijt}	when 1, captures the information about delays in a task with progress status $n = 0$
${}^{B}X_{ijt}$	the batch-size of delayed task with progress status $n = 0$
${}^{L}X_{ijt}$	yield-loss of delayed task with progress status $n = 0$
${}^{M}X_{ijt}$	maintenance-size of delayed maintenance task with progress status $n = 0$

References

1. Harjunkoski, I.; Maravelias, C.T.; Bongers, P.; Castro, P.M.; Engell, S.; Grossmann, I.E.; Hooker, J.; Méndez, C.A.; Sand, G.; Wassick, J.M. Scope for industrial applications of production scheduling models and solution methods. *Computers and Chemical Engineering* **2014**, *62*, 161–193.
2. Kelly, J.D.; Mann, J. Crude oil blend scheduling optimization : an application with multimillion dollar benefits. *Hydrocarbon Processing* **2003**.
3. Méndez, C.A.; Cerdá, J.; Grossmann, I.E.; Harjunkoski, I.; Fahl, M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers and Chemical Engineering* **2006**, *30*, 913–946.
4. Maravelias, C.T. General framework and modeling approach classification for chemical production scheduling. *AIChE Journal* **2012**, *58*, 1812–1828.
5. Velez, S.; Maravelias, C.T. Reformulations and branching methods for mixed-integer programming chemical production scheduling models. *Industrial and Engineering Chemistry Research* **2013**, *52*, 3832–3841.
6. Wassick, J.M.; Ferrio, J. Extending the resource task network for industrial applications. *Computers & Chemical Engineering* **2011**, *35*, 2124–2140.
7. Nie, Y.; Biegler, L.T.; Villa, C.M.; Wassick, J.M. Discrete Time Formulation for the Integration of Scheduling and Dynamic Optimization. *Industrial & Engineering Chemistry Research* **2015**, *54*, 4303–4315.
8. Gupta, D.; Maravelias, C.T.; Wassick, J.M. From rescheduling to online scheduling. *Chemical Engineering Research and Design* **2016**, *116*, 83–97.
9. Cott, B.J.; Macchietto, S. Minimizing the effects of batch process variability using online schedule modification. *Computers and Chemical Engineering* **1989**, *13*, 105–113.
10. Kanakamedala, K.B.; Reklaitis, G.V.; Venkatasubramanian, V. Reactive schedule modification in multipurpose batch chemical plants. *Industrial & Engineering Chemistry Research* **1994**, *33*, 77–90.
11. Huercio, A.; Espuña, A.; Puigjaner, L. Incorporating on-line scheduling strategies in integrated batch productioncontrol. *Computers & Chemical Engineering* **1995**, *19*, 609–614.
12. Kim, M.; Lee, I.B. Rule-based reactive rescheduling system for multi-purpose batch processes. *Computers & Chemical Engineering* **1997**, *21*, S1197–S1202.
13. Ko, D.; Na, S.; Moon, I.; Oh, M.; Dong-Gu Samsung, T.S. Development of a Rescheduling System for the Optimal Operation of Pipeless Plants. *Computers and Chemical Engineering* **1999**, *23*, S523–S526.
14. Huang, W.; Chung, P.W.H. A constraint approach for rescheduling batch processing plants including pipeless plants. *Computer Aided Chemical Engineering* **2003**, *14*, 161–166.
15. Henning, G.P.; Cerdá, J. Knowledge-based predictive and reactive scheduling in industrial environments. *Computers & Chemical Engineering* **2000**, *24*, 2315–2338.
16. Palombarini, J.; Martínez, E. SmartGantt - An interactive system for generating and updating rescheduling knowledge using relational abstractions. *Computers and Chemical Engineering* **2012**, *47*, 202–216.
17. Elkamel, A.; Mohindra, A. A rolling horizon heuristic for reactive scheduling of batch process operations. *Engineering Optimization* **1999**, *31*, 763–792.
18. Vin, J.; Ierapetritou, M.G. A new approach for efficient rescheduling of multiproduct batch plants. *Industrial and Engineering Chemistry Research* **2000**, *39*, 4228–4238.

19. Méndez, C.A.; Cerdá, J. Dynamic scheduling in multiproduct batch plants. *Computers and Chemical Engineering* **2003**, *27*, 1247–1259.
20. Ferrer-Nadal, S.; Méndez, C.A.; Graells, M.; Puigjaner, L. Optimal reactive scheduling of manufacturing plants with flexible batch recipes. *Industrial and Engineering Chemistry Research* **2007**, *46*, 6273–6283.
21. Janak, S.L.; Floudas, C.A.; Kallrath, J.; Vormbrock, N. Production scheduling of a large-scale industrial batch plant. II. Reactive scheduling. *Industrial and Engineering Chemistry Research* **2006**, *45*, 8253–8269.
22. Novas, J.M.; Henning, G.P. Reactive scheduling framework based on domain knowledge and constraint programming. *Computers and Chemical Engineering* **2010**, *34*, 2129–2148.
23. Honkomp, S.; Mockus, L.; Reklaitis, G.V. A framework for schedule evaluation with processing uncertainty. *Computers and Chemical Engineering* **1999**, *23*, 595–609.
24. Subramanian, K.; Maravelias, C.T.; Rawlings, J.B. A state-space model for chemical production scheduling. *Computers & Chemical Engineering* **2012**, *47*, 97–110.
25. Gupta, D.; Maravelias, C.T. On deterministic online scheduling: Major considerations, paradoxes and remedies. *Computers & Chemical Engineering* **2016**, *94*, 312–330.
26. Subramanian, K.; Rawlings, J.B.; Maravelias, C.T.; Flores-Cerrillo, J.; Megan, L. Integration of control theory and scheduling methods for supply chain management. *Computers and Chemical Engineering* **2013**, *51*, 4–20.
27. Subramanian, K.; Rawlings, J.B.; Maravelias, C.T. Economic model predictive control for inventory management in supply chains. *Computers and Chemical Engineering* **2014**, *64*, 71–80.
28. Kondili, E.; Pantelides, C.C.; Sargent, R.W.H. A general algorithm for short-term scheduling of batch operations-I. MILP formulation. *Computers and Chemical Engineering* **1993**, *17*, 211–227.
29. Pantelides, C.C. Unified frameworks for optimal process planning and scheduling. Proceedings on the second conference on foundations of computer aided operations; Publications, Cache: New York, 1994; pp. 253—-274.
30. Velez, S.; Maravelias, C.T. Advances in Mixed-Integer Programming Methods for Chemical Production Scheduling. *Annual Review of Chemical and Biomolecular Engineering* **2014**, *5*, 97–121.
31. Sundaramoorthy, A.; Maravelias, C.T. Computational Study of Network-Based Mixed-Integer Programming Approaches for Chemical Production Scheduling. *Industrial and Engineering Chemistry Research* **2011**, *50*, 5023–5040.
32. Pinto, J.M.; Grossmann, I.E. A Continuous Time Mixed Integer Linear Programming Model for Short Term Scheduling of Multistage Batch Plants. *Industrial and Engineering Chemistry Research* **1995**, *34*, 3037–3051.
33. Blomer, F.; Gunther, H.O. LP-based heuristics for scheduling chemical batch processes. *International Journal of Production Research* **2000**, *38*, 1029–1051.
34. Velez, S.; Maravelias, C.T. Mixed-integer programming model and tightening methods for scheduling in general chemical production environments. *Industrial and Engineering Chemistry Research* **2013**, *52*, 3407–3423.
35. Merchan, A.F.; Maravelias, C.T. Reformulations of Mixed-Integer Programming Continuous-Time Models for Chemical Production Scheduling. *Industrial & Engineering Chemistry Research* **2014**, *53*, 10155–10165.
36. Burkard, R.; Hatzl, J. Review, extensions and computational comparison of MILP formulations for scheduling of batch processes. *Computers & Chemical Engineering* **2005**, *29*, 1752–1769.
37. Janak, S.L.; Floudas, C.A. Improving unit-specific event based continuous-time approaches for batch processes: Integrality gap and task splitting. *Computers & Chemical Engineering* **2008**, *32*, 913–955.
38. Lee, H.; Maravelias, C.T. Discrete-time mixed-integer programming models for short-term scheduling in multipurpose environments. *Computers & Chemical Engineering* **2017**.
39. Sahinidis, N.; Grossmann, I. Reformulation of multiperiod MILP models for planning and scheduling of chemical processes. *Computers & Chemical Engineering* **1991**, *15*, 255–272.
40. Yee, K.; Shah, N. Improving the efficiency of discrete time scheduling formulation. *Computers & Chemical Engineering* **1998**, *22*, S403–S410.
41. Lee, H.; Maravelias, C.T. Mixed-integer programming models for simultaneous batching and scheduling in multipurpose batch plants. *Computers & Chemical Engineering* **2017**, *106*, 621–644.
42. Papageorgiou, L.G.; Pantelides, C.C. Optimal campaign planning/scheduling of multipurpose batch/semicontinuous plants. 2. A mathematical decomposition approach. *Industrial and Engineering Chemistry Research* **1996**, *35*, 510–529.

43. Bassett, M.H.; Pekny, J.F.; Reklaitis, G.V. Decomposition techniques for the solution of large-scale scheduling problems. *AIChE Journal* **1996**, *42*, 3373–3387.
44. Kelly, J.D.; Zyngier, D. Hierarchical decomposition heuristic for scheduling: Coordinated reasoning for decentralized and distributed decision-making problems. *Computers & Chemical Engineering* **2008**, *32*, 2684–2705.
45. Wu, D.; Ierapetritou, M.G. Decomposition approaches for the efficient solution of short-term scheduling problems. *Computers & Chemical Engineering* **2003**, *27*, 1261–1276.
46. Calfa, B.A.; Agarwal, A.; Grossmann, I.E.; Wassick, J.M. Hybrid Bilevel-Lagrangean Decomposition Scheme for the Integration of Planning and Scheduling of a Network of Batch Plants. *Industrial & Engineering Chemistry Research* **2013**, *52*, 2152–2167.
47. Castro, P.M.; Harjunkoski, I.; Grossmann, I.E. Greedy algorithm for scheduling batch plants with sequence-dependent changeovers. *AIChE Journal* **2011**, *57*, 373–387.
48. Roslöf, J.; Harjunkoski, I.; Björkqvist, J.; Karlsson, S.; Westerlund, T. An MILP-based reordering algorithm for complex industrial scheduling and rescheduling. *Computers & Chemical Engineering* **2001**, *25*, 821–828.
49. Kopanos, G.M.; Méndez, C.A.; Puigjaner, L. MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. *European Journal of Operational Research* **2010**, *207*, 644–655.
50. Relvas, S.; Barbosa-Póvoa, A.P.F.; Matos, H.A. Heuristic batch sequencing on a multiproduct oil distribution system. *Computers & Chemical Engineering* **2009**, *33*, 712–730.
51. Jain, V.; Grossmann, I.E. Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. *INFORMS Journal on Computing* **2001**, *13*, 258–276.
52. Harjunkoski, I.; Grossmann, I.E. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers & Chemical Engineering* **2002**, *26*, 1533–1552.
53. Maravelias, C.T.; Grossmann, I.E. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Computers & Chemical Engineering* **2004**, *28*, 1921–1949.
54. Roe, B.; Papageorgiou, L.G.; Shah, N. A hybrid MILP/CLP algorithm for multipurpose batch process scheduling. *Computers & Chemical Engineering* **2005**, *29*, 1277–1291.
55. Maravelias, C.T. A decomposition framework for the scheduling of single- and multi-stage processes. *Computers & Chemical Engineering* **2006**, *30*, 407–420.
56. Subrahmanyam, S.; Kudva, G.K.; Bassett, M.H.; Pekny, J.F. Application of distributed computing to batch plant design and scheduling. *AIChE Journal* **1996**, *42*, 1648–1661.
57. Ferris, M.C.; Maravelias, C.T.; Sundaramoorthy, A. Simultaneous Batching and Scheduling Using Dynamic Decomposition on a Grid. *INFORMS Journal on Computing* **2009**, *21*, 398–410.
58. Velez, S.; Maravelias, C.T. A branch-and-bound algorithm for the solution of chemical production scheduling MIP models using parallel computing. *Computers & Chemical Engineering* **2013**, *55*, 28–39.
59. Shah, N.; Pantelides, C.C.; Sargent, R.W.H. A general algorithm for short-term scheduling of batch operations-II. Computational issues. *Computers and Chemical Engineering* **1993**, *17*, 229–244.
60. Stephanopoulos, G. *Chemical Process Control : An Introduction to Theory and Practice*; Prentice-Hall, 1984; p. 696.
61. Ogunnaike, B.A.; Ray, W.H. *Process Dynamics, Modeling, and Control*; Oxford University Press, 1994; p. 1260.
62. Bequette, B.W. *Process Control : Modeling, Design, and Simulation*; Prentice Hall PTR, 2003; p. 769.
63. Rawlings, J.B.; Mayne, D. *Model Predictive Control : Theory and Design*; Nob Hill Pub, 2009; p. 669.
64. Seborg, D.E.; Edgar, T.F.; Duncan, M.A.; Doyle III, F.J. *Process Dynamics and Control*; Wiley, 2016; p. 502.
65. Amrit, R.; Rawlings, J.B.; Biegler, L.T. Optimizing process economics online using model predictive control. *Computers & Chemical Engineering* **2013**, *58*, 334–343.
66. Ellis, M.; Durand, H.; Christofides, P.D. A tutorial review of economic model predictive control methods. *Journal of Process Control* **2014**, *24*, 1156–1178.
67. Rawlings, J.B.; Risbeck, M.J. Model predictive control with discrete actuators: Theory and application. *Automatica* **2017**, *78*, 258–265.
68. Baldea, M.; Harjunkoski, I. Integrated production scheduling and process control: A systematic review. *Computers and Chemical Engineering* **2014**, *71*, 377–390.

69. Li, Z.; Ierapetritou, M.G. Process scheduling under uncertainty: Review and challenges. *Computers and Chemical Engineering* **2008**, *32*, 715–727.
70. Janak, S.L.; Lin, X.; Floudas, C.A. A new robust optimization approach for scheduling under uncertainty. II. Uncertainty with known probability distribution. *Computers and Chemical Engineering* **2007**, *31*, 171–195.
71. Sand, G.; Engell, S. Modeling and solving real-time scheduling problems by stochastic integer programming. *Computers and Chemical Engineering* **2004**, *28*, 1087–1103.
72. Sabuncuoglu, I.; Karabuk, S. Rescheduling frequency in an fms with uncertain processing times and unreliable machines. *Journal of Manufacturing Systems* **1999**, *18*, 268–283.
73. Chaari, T.; Chaabane, S.; Aissani, N.; Trentesaux, D. Scheduling under uncertainty: Survey and research directions. 2014 International Conference on Advanced Logistics and Transport, ICALT 2014, 2014, pp. 229–234.
74. Martagan, T.; Krishnamurthy, A. Control and Optimization of Bioprocesses Using Markov Decision Process. Proceedings of the 2012 Industrial and Systems Engineering Research Conference, 2012, number 1.
75. Martagan, T.; Krishnamurthy, A.; Maravelias, C.T. Optimal condition-based harvesting policies for biomanufacturing operations with failure risks. *IIE Transactions* **2016**, *48*, 440–461.
76. Dedopoulos, I.T.; Shah, N. Optimal Short-Term Scheduling of Maintenance and Production for Multipurpose Plants. *Industrial and Engineering Chemistry Research* **1995**, *34*, 192–201.
77. Sanmartí, E.; Espuña, A.; Puigjaner, L. Batch production and preventive maintenance scheduling under equipment failure uncertainty. *Computers & Chemical Engineering* **1997**, *21*, 1157–1168.
78. Vassiliadis, C.; Pistikopoulos, E. Maintenance scheduling and process optimization under uncertainty. *Computers & Chemical Engineering* **2001**, *25*, 217–236.
79. Kopanos, G.M.; Xenos, D.P.; Cicciotti, M.; Pistikopoulos, E.N.; Thornhill, N.F. Optimization of a network of compressors in parallel: Operational and maintenance planning – The air separation plant case. *Applied Energy* **2015**, *146*, 453–470.
80. Xenos, D.P.; Kopanos, G.M.; Cicciotti, M.; Thornhill, N.F. Operational optimization of networks of compressors considering condition-based maintenance. *Computers & Chemical Engineering* **2016**, *84*, 117–131.
81. Biondi, M.; Sand, G.; Harjunkoski, I. Optimization of multipurpose process plant operations: A multi-time-scale maintenance and production scheduling approach. *Computers & Chemical Engineering* **2017**, *99*, 325–339.