Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-7 (2022) 821–826

The Similarity Index to Decompose Two-Stage Stochastic Scheduling Problems

Daniel Montes * **José Luis Pitarch** ** **Cesar de Prada** *,***

* *Department of Systems Engineering and Automatic Control, Universidad de Valladolid, Spain (e-mail: danielalberto.montes.lopez@uva.es, prada@autom.uva.es).*

** *Instituto U. de Automática e Informática Industrial (ai2), Universitat Politècnica de Valencia, Spain (e-mail: jlpitarch@isa.upv.es).*

*** *Institute of Sustainable Processes, Universidad de Valladolid, Spain*

Abstract: Two-stage stochastic scheduling problems often involve a large number of continuous and discrete variables, so finding solutions in short time periods is challenging and computationally expensive. However, for online or closed-loop scheduling implementations, optimal or near-optimal solutions are required in real-time. We propose a decomposition method based on the so-called Similarity Index (SI). An iterative procedure is set up so that each sub-problem (corresponding to a scenario) is solved independently, aiming to optimize the original cost function while maximizing the similarity of the first-stage variables among the scenarios solutions. The SI is incorporated into each subproblem cost function, multiplied by a penalty parameter that is updated in each iteration until reaching complete similarity in the first-stage variables among all subproblems. The method is applied to schedule production and maintenance tasks in an evaporation network. The tests show that significant benefits are expected in terms of computational demands as the number of scenarios increases.

Keywords: Decomposition, Online Scheduling, Uncertainty, Mixed-Integer Optimization

1. INTRODUCTION

Uncertainty is always present in process operation. It can come from many sources but the consequences are clear: prediction errors and suboptimal, or even infeasible, operation (Sahinidis, 2004). Multi-stage stochastic formulations take uncertainty into account by discretizing the plausible uncertainty realizations. This work focuses on two-stage formulations, where the problem decisions are split in two: first-stage decisions, which are taken with the information available a priori and cannot be changed later; and second-stage ones, which can be taken later on according to the actual uncertainty realization. The time periods corresponding to the first-stage variables are said to be within the robust horizon. When applied to scheduling problems, large-scale models that involve both continuous and discrete decisions result. Hence, multi-stage scheduling problems are computationally challenging and difficult to solve to optimality, whilst industrial practice often requires good solutions in short time periods.

A common approach for dealing with such large-scale problems is to divide them into smaller subproblems, defined by local constraints, which can be solved in parallel. The solutions to those subproblems are used for updating a global master problem defined by global constraints. Both the master and the subproblems are alternately solved until some termination criterion is met. Benders Decomposition (BD) and the Progressive Hedging Algorithm (PHA) are the most often used decomposition methods for solving two-stage stochastic scheduling problems.

BD consists of three consecutive steps: projection, dualization, and relaxation (Benders, 1962). The base method produces a weak master problem formulation that loses all relevant information about the second-stage variables. This leads to slow convergence of the procedure in the end. A cut is generated in each iteration, depending on whether the subproblems are optimal or infeasible. Thus, BD has a high computational burden and it is often outperformed by the monolithic formulation. The available enhanced strategies for overcoming the BD drawbacks can be classified into four categories: decomposition strategy, solution generation, cut generation, and solution procedure. See (Rahmaniani et al., 2017) for a thorough review. In particular, for two-stage scheduling problems, partial BD allows the master problem to retain information of all scenario subproblems, which reduces the number of generated cuts, increasing thus convergence speed (Crainic et al., 2016).

The PHA consists in relaxing the non-anticipativity constraints so that they are incorporated into the cost functions associated with a Lagrange-like multiplier (or penalty parameter) that is updated iteratively until a solution is found (Rockafellar and Wets, 1991). It results in a full decomposition over the scenarios while progressively enforcing the non-anticipativity constraints. The PHA has been used successfully to obtain high-quality solutions to stochastic scheduling problems (Gade et al., 2016; Huang and Zheng, 2020). However, there are still open concerns when dealing with large-scale mixed-integer problems: the multiplier-update rule, choice of termination criteria, and techniques for accelerating convergence (Watson and Woodruff, 2010).

10.1016/j.ifacol.2022.07.546

In this work, we present an alternative decomposition method for two-stage scheduling problems based on a similarity index among first-stage discrete decisions. This method enables full scenario decomposition with a single tuning parameter. The similarity index aggregates the solution of the first-stage discrete variables into a single number. Then, the scenario subproblems incorporate such index in their costs functions so that the similarity among the other scenarios is maximized. Moreover, the similarity index is also used as a termination criterion, as it must reach the value of 1 to fulfill non-anticipativity constraints. The proposed decomposition method is tested to schedule operations in an evaporation network. Optimal solutions are found and convergence is achieved in a few iterations, without using acceleration techniques or problem-dependent heuristics.

The paper is organized as follows: Section 2 describes the proposed decomposition method; Section 3 describes the case study adapted from the literature; the results of the monolithic and decomposed problems are discussed in Section 4 and; finally, Section 5 closes the paper with some remarks and an outline for next steps.

2. DESCRIPTION OF THE METHOD

The concept of similarity index (SI), first presented in Palacin et al. (2017), is inspired by the idea of minimum agreement index (Sakawa and Kubota, 2000). It measures the robustness of the solution in two-stage scheduling problems, by merging the information on how similar the recourse variables computed for each scenario are in a single indicator. In this way, a SI equal to 1 means that the schedules for all scenarios are identical, i.e., the risk-averse solution. The SI is computed by fuzzifying the discrete decisions taken within a prediction horizon. A discrete decision in a scenario is weighted by 1 (100%) if taken on a particular time instant t, but it is also accounted for the near time instants t_n with decreasing values that are proportional to the time differences $|t - t_n|$. Then, the SI is computed as the intersection area among all scenarios, divided by the total possible area (the area if all the scenario solutions coincide). Figure 1 shows a graphical representation of the SI computation in an example where the time horizon is discretized and discrete decisions are fuzzified along 7 time periods.

Formally, the SI formula defined in Palacin et al. (2017) considered a three-period span to fuzzify the discrete decisions:

$$SI := \sum_{t \in \mathcal{M}_u \setminus \{t_F\}} \frac{\min\limits_{e \in \mathcal{E}} \left\{ y_{te} + 0.5 y_{(t+1)e} + 0.5 y_{(t-1)e} \right\}}{2\,(n_u - 1) + 1.5} \quad (1)$$

Where $\mathcal{E}$ is the scenario set, $\mathcal{M}_u$ is the subset that contains the time periods that do not belong to the robust horizon (i.e., those allowing recourse decisions), t_F is the horizon end, $y_{te} \in \{0, 1\}$ is the set of possible discrete decisions from which only one is to be scheduled at time t and scenario e, and n_u the number of time periods in $\mathcal{M}_u$.

Note that the numerator of (1), i.e. the intersection area, involves a nonlinear operation ($\min_e$). Therefore, to be used in linear scheduling formulations, a lower bound on (1) can be formulated by introducing a set of slack variables

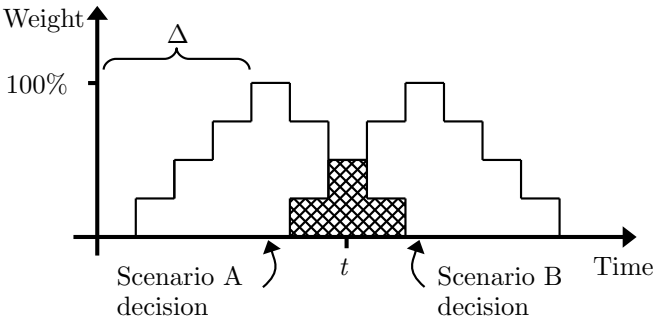

Fig. 1. Example of fuzzifying a discrete decision along $\pm\Delta$ time periods. The SI is computed from the intersection between scenarios A and B and the total fuzzified area.

($s_t \in \mathbb{R}^+$) and inequalities as shown in (2). In this way, the lower bound on SI would be computed by (3). For this lower bound to be tight, the slack variables need to be included in the objective function so that they are maximized. Hence, the slack variables s_t will take the higher value that constraints (2) allow.

$$s_t \leq y_{te} + 0.5 y_{(t+1)e} + 0.5 y_{(t-1)e} \; \forall t \in \mathcal{M}_u \setminus \{t_F\}, e \in \mathcal{E} \quad (2)$$

$$SI \geq \sum_{t \in \mathcal{M}_u \setminus \{t_F\}} \frac{s_t}{2\,(n_u - 1) + 1.5} \quad (3)$$

2.1 Application to the robust horizon

Although the similarity index was conceived as a means of specifying the desired robustness level in stochastic scheduling formulations, there is no reason to restrict its use to second-stage variables. If applied to first-stage variables in the solution, it will be always 1, as all scenario solutions coincide for such variables by definition. However, if the non-anticipativity constraints are removed from the formulation, a scheduling solution can be computed independently for each scenario, but such solutions would not certainly coincide in the first stage, and the SI would take values less than 1. Nonetheless, an iterative procedure could be set up in this situation to progressively push the SI forward up to the point where the solution of all scenarios is equal along the robust horizon. That is the proposal of this paper: a method to achieve full scenario decomposition in two-stage stochastic scheduling problems by just using the similarity index. This is accomplished by completely ignoring the non-anticipativity constraints in the formulation instead of relaxing them as in the PH algorithm (Rockafellar and Wets, 1991).

The general formulas to compute the SI and its lower bound for first-stage variables are (4)-(6).

$$SI := \sum_{t=1}^{t_R} \min_{e \in \mathcal{E}} \Big\{ y_{te} + \sum_{\tau=1}^{\Delta} \frac{\Delta - \tau}{\Delta} (y_{(t-\tau)e}|_{t>\tau} + y_{(t+\tau)e}|_{t+\tau \leq t_R}) \Big\} \Big/ \Big(t_R \Delta - 2 \sum_{\tau=1}^{\Delta} \tau \frac{\Delta - \tau}{\Delta} \Big) \quad (4)$$

$$s_t \leq y_{te} + \sum_{\tau=1}^{\Delta} \frac{\Delta - \tau}{\Delta} (y_{(t-\tau)e}|_{t>\tau} + y_{(t+\tau)e}|_{t+\tau \leq t_R}) \quad \forall e \in \mathcal{E};\; t : 1, \ldots, t_R \quad (5)$$

$$SI \geq \sum_{t=1}^{t_R} s_t \Big/ \Big(t_R \Delta - 2 \sum_{\tau=1}^{\Delta} \tau \frac{\Delta - \tau}{\Delta} \Big) \quad (6)$$

In the above formulas, Δ is the number of time periods along which the discrete decisions are fuzzified (see Fig. 1), t_R is the end of the robust horizon and, with some abuse of notation, $y_{(t-\tau)}|_{t>\tau}$ means that $y_{(t-\tau)}$ is only accounted in the summation if $t > \tau$. Note that these formulas are slightly different from the particular case shown in (1)-(3), as both extremes of the robust horizon (those within Δ periods) need special treatment in the fuzzification process.

Remark 1. The nice feature of the SI is that it is capable of informing on the fulfillment of the non-anticipativity constraints in a single value, no matter how long the robust horizon is nor the number of complicating first-stage variables.

2.2 Decomposition algorithm

The core idea of the method is to solve each one of the scenarios (subproblems) independently and then compare their solutions using the similarity index. An iterative procedure is set up so that the SI is progressively improved. For this, several SI need to be computed: local ones SI_e referred to each scenario, which are part of the optimization subproblems; and a global one SI that is computed after the optimization solutions have been collected. Note that each subproblem needs information from the others to compute its SI_e (see constraints (2)), but only the first-stage variables corresponding to the current scenario can be modified by the optimizer. A possible approach to fix the values for the rest of the scenarios would be to use the solutions got by each subproblem in the previous iteration. However, this could lead to situations where a subproblem cannot improve its solution to get a higher similarity index because the first-stage values for the other scenarios fixed in the previous iteration are too different. If this situation gets entrenched and no local problem can improve its SI_e, progress stops, and the global problem gets stuck in an infeasible solution: the global SI never reaches 1, so non-anticipativity does not hold.

A better approach to overcome this drawback is to compute SI_e only with the variables of the current scenario plus the ones fixed by a single solution computed somehow from the previous iteration. For instance, the solution of the scenario that reported the *worst* local SI in the previous iteration. This way, if a subproblem cannot move its solution further to improve the SI due to particular hard constraints (high production, extreme ambient conditions, etc.), the others will move to adjust it.

As mentioned earlier, the local SI calculation requires the maximization of the slack variables s_t. Consequently, the objective functions of the subproblems need an additional term to account for it (negative sign when minimizing, positive otherwise). This term can be the SI lower bound (6) itself, times a multiplier λ that is updated in each iteration. In this way, the optimizer gives more importance to maximizing the SI as the iterations increase.

We propose an update rule for λ inspired by the subgradient method (Shor, 1985):

$$\lambda_{k+1} = \lambda_k - \alpha_{k+1}\,(SI - 1) \tag{7}$$

Where α is a tuning parameter that decreases in each iteration k according to a user-defined factor, for example, $\alpha_{k+1} = 0.9\alpha_k$. This means that, in the beginning, the multiplier λ would presumably be small, so more preference is given to the original objective function J in order to accelerate the procedure towards an optimal solution. Afterward, λ progressively increases to fulfill the non-anticipativity feature as close as possible to that optimal solution (Bazaraa and Sherali, 1981). As the iteration count increases, more importance will be given to maximizing the SI. The algorithm stops either after a maximum number of iterations or when the global SI reaches 1. The reader is referred to Algorithm 1 for a more succinct explanation. The local optimization problem for each scenario e would then read as:

$$\begin{aligned}
\min_{y_{te},\, s_t} \quad & J_e - \lambda \sum_{t=1}^{t_R} s_t \Big/ \Big(t_R \Delta - 2 \sum_{\tau=1}^{\Delta} \tau \frac{\Delta - \tau}{\Delta} \Big) \\
\text{s.t.} \quad & \\
& \text{Local model constraints} \qquad (8) \\
& s_t \le \bar{y}_t + \sum_{\tau=1}^{\Delta} \frac{\Delta - \tau}{\Delta} \big(\bar{y}_{(t-\tau)}|_{t>\tau} + \bar{y}_{(t+\tau)}|_{t+\tau \le t_R} \big) \\
& s_t \le y_{te} + \sum_{\tau=1}^{\Delta} \frac{\Delta - \tau}{\Delta} \big(y_{(t-\tau)e}|_{t>\tau} + y_{(t+\tau)e}|_{t+\tau \le t_R} \big) \\
& \qquad\qquad\qquad\qquad t : 1, \ldots, t_R
\end{aligned}$$

Where $\bar{y}_t$ refers to the solution of the scenario showing the worst local SI in the previous iteration. Note that the slack variables s_t are also tailored to each subproblem, and they are bounded by the local decision variables y_{te} as well as by those fixed from the previous iteration $\bar{y}_t$. Consequently, the local SI only compares the solution of the respective subproblem with the one that yielded the worst SI in the previous iteration.

Algorithm 1. Similarity Index Decomposition

Require: $\alpha_0, \mathrm{k}_{\max}, \Delta$

$k \leftarrow 0, \lambda_0 \leftarrow 0, \bar{y}_t \leftarrow 0$ ▷ Initialization
repeat
  for e in $\mathcal{E}$ **do**
    $y^*_{te}, s^*_t \leftarrow \arg\min_{y_{te}, s_t} J_e - \lambda_k SI_e$ ▷ Solve (8)
    $SI^*_e \leftarrow SI_e(s^*_t)$
  end for
  $SI \leftarrow$ Eq. (4) with y^*_{te} ▷ Global SI computation
  $\bar{y}_t \leftarrow \arg\min_{y^*_{te}} \{SI^*_e\}$ ▷ Solution with worst SI_e
  $\alpha_{k+1} \leftarrow 0.9\alpha_k$ ▷ Multiplier update
  $\lambda_{k+1} \leftarrow \lambda_k - \alpha_{k+1}\,(SI - 1)$
  $k \leftarrow k + 1$
until $SI = 1 \vee k = \mathrm{k}_{\max}$ ▷ Convergence check
return y^*_{te}

The algorithm has three parameters whose values need to be chosen: α_0 together with its decrease rate (suggested here to be 0.9) and the length of the fuzzifying horizon Δ. A right choice looks problem-dependent. Δ must be shorter than the length of the robust horizon t_R, but we do not recommend values over 3 due to computational reasons. In general, α_0 should be big enough so that $\alpha_k(SI - 1)$ does not vanish in a couple of iterations. However, large values favor convergence speed over optimality, meaning that a feasible solution is quickly reached but is probably suboptimal.

3. EVAPORATION NETWORK PROBLEM

In this section, we present a realistic industrial case study, a mixed production, and maintenance scheduling problem of an evaporation network, in which we tested the proposed decomposition method. The evaporation network aims to concentrate solvents that are used in the main process of a cellulose fiber factory. Each plant in the network consists of two evaporation chambers in serial connection with heat exchangers in between. Some plants show different nominal efficiencies. In addition, the evaporation plants need to be cleaned regularly, as fouling increases the steam consumption and so the processing costs.

The scheduler has several functions: link products to plants, assign loads, decide when to clean, stop and start-up plants. All of this is subject to several operational constraints, such as (uncertain) processing demands, each plant can only handle one product at a time, there is personnel to clean a single plant per day, (uncertain) weather forecast, etc. Palacín et al. (2018) addressed this kind of problem by a two-stage stochastic scheduling MILP formulation, that was based on predefined-precedence allocation. Such a mathematical model is replicated in the next section with some simplifications for the sake of clarity. Figure 2 shows the possible states and transitions for an evaporation plant: A plant that is under normal operation can either go to the cleaning stage or standby; a plant that has not been cleaned and is on standby can only go to the cleaning stage; a clean plant can be put on standby or into normal operation.

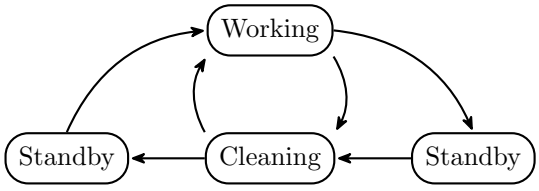

Fig. 2. State diagram of an evaporation plant.

3.1 Scheduling formulation

The original formulation of Palacín et al. (2018) represented the evolution of the fouling state in each plant by discretizing the time horizon in days and applying the concepts of finite-state machine and timed automaton (Behrmann et al., 2005). In this way, the resulting scheduling model includes the fouling dynamics, and the predefined-precedence approach saves an important number of binary variables that would be required in a general-precedence formulation. Nevertheless, for this paper, we decided to simplify the model flexibility (hence, its size), by replacing the original fouling-state discretization with an integer variable that informs on the number of days a plant has been in operation since the last cleaning. The number of possible plant stages is then reduced to just the four in Fig. 2: pre-cleaning and post-cleaning standby, cleaning, and working. This reduces the variables count by a factor of seven. However, this simplification comes at a cost: it is not possible to account for different types of cleaning and there is no way to specify a set of initial days in which cleaning is not certainly worth it.

Five different sets are defined for the model:

- $\mathcal{V}$ represents the set of evaporation plants.
- $\mathcal{S}$ denotes the possible plant states: s_G for operating stages, s_P for pre-cleaning standby stages, s_L for cleaning stages, and s_{PL} for post-cleaning standby stages.
- $\mathcal{M}$ is the set of t days in the discretized scheduling horizon. The first t_R days form the robust horizon $\mathcal{M}_R \subset \mathcal{M}$.
- $\mathcal{P}$ is the set of the ρ products to allocate.
- $\mathcal{E}$ represents the set of considered scenarios.

The variables that relate to these sets are:

- W_{vtse} binary variables that specify if plant v is at a stage s at day t in scenario e.
- P_{vtpe} binary variables that link product p to plant v at day t in scenario e.
- F_{vtpe} nonnegative variables to assign the evaporation flow of product p in plant v at day t in scenario e.
- D_{vte} nonnegative integer variables that state how many days a plant v has been in operation at day t in scenario e.

The cost C of an evaporation plant v processing a product p in each scenario e is approximated as a function of the assigned load F, the ambient temperature T_{out}, and the time in operation since the last cleaning D:

$$C(v,t,p) = \left(K_T(v)T_{out}(t) + K_E(v)\right) F_{vtp} + K_F(v)D_{vt} \quad (9)$$

Where $K_T(v)$ and $K_E(v)$ represent the nominal efficiencies of the different equipment in the evaporation plant, and $K_F(v)$ is the extra cost associated with the fouling state.

The feasible transitions between stages are formulated according to linear generalized disjunctive programming (Sawaya and Grossmann, 2005) following positive logic statements:

1. A plant can only be in a single state at a day.
$$\bigvee_{s\in\mathcal{S}} W_{vtse} \quad \forall v \in \mathcal{V}, \forall t \in \mathcal{M}, \forall e \in \mathcal{E} \quad (10)$$
2. A plant can only process a single product p at a time (if it is working).
$$\bigvee_{p\in\mathcal{P}} P_{vtpe} \vee W_{vts_Le} \vee W_{vts_Pe} \vee W_{vts_{PL}e} \quad \forall v \in \mathcal{V}, \forall t \in \mathcal{M}, \forall e \in \mathcal{E} \quad (11)$$
3. Only a single cleaning task can occur per day.
$$\bigvee_{v\in\mathcal{V}} W_{vts_Le} \quad \forall t \in \mathcal{M}, \forall e \in \mathcal{E} \quad (12)$$
4. A plant that has been put on standby without cleaning, can either be cleaned or remain on standby.
$$W_{vts_Pe} \rightarrow W_{v(t+1)s_Pe} \vee W_{v(t+1)s_Le} \quad \forall v \in \mathcal{V}, \forall t \in \mathcal{M} \setminus \{t_F\}, \forall e \in \mathcal{E} \quad (13)$$
5. After a cleaning task, a plant can either be put on standby or start operation.
$$W_{vts_Le} \rightarrow W_{v(t+1)s_{PL}e} \vee W_{v(t+1)s_Ge} \quad \forall v \in \mathcal{V}, \forall t \in \mathcal{M} \setminus \{t_F\}, \forall e \in \mathcal{E} \quad (14)$$
6. A plant that has been cleaned and is on standby, can remain in such a state or begin to operate.
$$W_{vts_{PL}e} \rightarrow W_{v(t+1)s_{PL}e} \vee W_{v(t+1)s_Ge} \quad \forall v \in \mathcal{V}, \forall t \in \mathcal{M} \setminus \{t_F\}, \forall e \in \mathcal{E} \quad (15)$$

7. No changes in the product-plant allocation are allowed until the plant is cleaned.

$$P_{vtpe} \rightarrow P_{v(t+1)pe} \vee W_{v(t+1)se} \; \forall v \in \mathcal{V}, \forall t \in \mathcal{M} \setminus \{t_F\}, s \in \{s_P, s_L\}, \forall p \in \mathcal{P}, \forall e \in \mathcal{E} \quad (16)$$

8. To avoid infeasibility in the long run, no plants are allowed to end up on standby before cleaning.

$$\neg W_{vt_F s_P e} \quad \forall v \in \mathcal{V}, \forall e \in \mathcal{E} \quad (17)$$

9. A terminal cost is established to account for the plants that end up in an advanced fouling state.

$$T_C := 0.5 K_L \; W_{vt_F s_G e} \quad \forall v \in \mathcal{V}, \forall e \in \mathcal{E} \quad (18)$$

Where K_L represents the cost of a cleaning task.

10. D_{vte} increase as long as plant v is in operation, hold if the plant is stopped, and reset to zero when cleaning.

$$\begin{bmatrix} \neg W_{vts_L e} \\ D_{vte} = D_{v(t-1)e} + W_{vts_G e} \end{bmatrix} \vee \begin{bmatrix} W_{vts_L e} \\ D_{vte} = 0 \end{bmatrix} \quad \forall v \in \mathcal{V}, e \in \mathcal{E}, \; t \in \mathcal{M} \setminus \{t_1\} \quad (19)$$

Additionally, the model has some production constraints:

- The evaporation rate in each plant is bounded (unless the plant is not working, in which case it is zero). The lower limit is fixed while the upper limit has a known dependency on the ambient temperature.

$$\begin{bmatrix} P_{vtpe} \\ L_v \leq F_{vtpe} \\ F_{vtpe} \leq U_v\,(T_{out}) \end{bmatrix} \vee \begin{bmatrix} \neg P_{vtpe} \\ F_{vtpe} = 0 \end{bmatrix} \begin{matrix} \forall v \in \mathcal{V}, \forall t \in \mathcal{M} \\ \forall p \in \mathcal{P}, \forall e \in \mathcal{E} \end{matrix} \quad (20)$$

- The production demands SP_{pte} for each product must be accomplished every day.

$$\sum_{v \in \mathcal{V}} F_{vtpe} = SP_{pte} \quad \forall t \in \mathcal{M}, \forall p \in \mathcal{P}, \forall e \in \mathcal{E} \quad (21)$$

The non-anticipativity constraints are enforced during t_R days.

$$W_{vtse} \equiv W_{vts}, \quad P_{vtpe} \equiv P_{vtp}, \quad F_{vtpe} \equiv F_{vtp} \quad \forall v \in \mathcal{V}, \; t \in \mathcal{M}_R, \; s \in \mathcal{S}, \; p \in \mathcal{P}, \; e \in \mathcal{E} \quad (22)$$

The remaining constraints are the known initial state of the plants, denoted by the set $\mathcal{I}$, and the allowed connections between products and plants, denoted by $\mathcal{A}$. Then, the stochastic production-maintenance scheduling problem reads as:

$$\begin{aligned} \min \quad & \frac{1}{2^{\rho+1}|\mathcal{M}|} \sum_{e \in \mathcal{E}} \sum_{t \in \mathcal{M}} \sum_{v \in \mathcal{V}} \Big[K_F(v) D_{vte} + K_L W_{vts_L e} \\ & + \sum_{p \in \mathcal{P}} \left(K_T(v) T_{out}(t) + K_E(v) \right) F_{vtpe} \Big] + T_C \\ \text{s.t.} \quad & \text{Constraints } (10) - (22) \\ & \{W_{v1se}, P_{v1se}, D_{v1e}, F_{v1pe}\} \in \mathcal{I} \\ & P_{vtpe} \in \mathcal{A}, \quad F_{vtpe} \in \mathbb{R}_{\geq 0}, \quad D_{vte} \in \mathbb{N} \\ & \{W_{vtse}, P_{vtpe}\} \in \{0, 1\} \end{aligned} \quad (23)$$

See (Palacín et al., 2018; Palacín, 2020) for a detailed description of the case study and the non-simplified model.

3.2 Adaptation to decomposition

The decomposition method requires slight modifications to the model. The non-anticipativity constraints (22) are disregarded, as schedules for scenarios are to be solved independently via (8).

Similarity index calculation. As described in Section 2, local SI_e need to be calculated as part of the decomposition method. Then, after all the solutions have been collected, a global SI is computed to update the lagrangian-like multiplier λ. Considering $\Delta = 2$, the local SI_e depend on the decision variables defined for each scenario as follows:

$$SI_e = \sum_{v \in \mathcal{V}} \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{M}_R} \frac{s_{vts}}{n_v\,(2t_R - 1)} \quad (24)$$

$$\begin{aligned} s_{vts} &\leq \overline{W}_{vts} + \frac{1}{2}\left(\overline{W}_{v(t-1)s}|_{t>1} + \overline{W}_{v(t+1)s}|_{t+1 \leq t_R}\right) \\ s_{vts} &\leq W_{vtse} + \frac{1}{2}\left(W_{v(t-1)se}|_{t>1} + W_{v(t+1)se}|_{t+1 \leq t_R}\right) \\ & \qquad \forall v \in \mathcal{V}, s \in \mathcal{S}, t \in \mathcal{M}_R \end{aligned} \quad (25)$$

Where n_v is the number of evaporation plants and $\overline{W}_{vts}$ is the previous solution for the scenario that yielded the worst SI. Note that variables P_{vtpe} should also be included in the SI computation. However, this can be neglected due to the particular nature of the case study: if the plants state is the same for all scenarios within the robust horizon, there is no possibility that a plant is optimally allocated to different products among scenarios because the evaporation demand is not uncertain within the robust horizon.

Following (4), the global SI is calculated by:

$$SI = \sum_{v \in \mathcal{V}} \sum_{s \in \mathcal{S}} \sum_{t \in \mathcal{M}_R} \min_{e \in \mathcal{E}} \Big\{ W_{vtse} + \frac{1}{2}\big(W_{v(t-1)se}|_{t>1} + W_{v(t+1)se}|_{t+1 \leq t_R}\big) \Big\} \Big/ \Big(n_v\,(2t_R - 1) \Big) \quad (26)$$

4. RESULTS AND DISCUSSION

Two problems instances were studied, both with three plants and two products to allocate. The scheduling horizon is $t_F = 30$ and the robust horizon is $t_R = 7$. The only difference is the number of scenarios considered for the expected production demand: four and eight. The problem instance with four scenarios has 3243 continuous variables and 2160 binary variables. The other instance has twice as many variables.

All model instances were solved in GAMS 36.2.0 using IBM ILOG CPLEX 20.1 with an i7–1185G7 CPU at 3.00GHz and 16GB of RAM. The relative gap tolerance was set to zero. The monolithic problem was solved in parallel with four threads using the solver default capabilities. In the case of the decomposed problem, the subproblems were solved in parallel using GAMS grid facility with two threads assigned to each one. α was initially set to 5000 and it was reduced by 3% in each iteration.

For the first instance, the solutions of both the monolithic and the decomposed problems are identical, and their cost functions are equal to 10174.8¤. The monolithic approach elapsed **17.96** CPU seconds, while the decomposed approach was around 4% faster, elapsing **17.25** s and four iterations to converge. Table 1 reports the evolution of the global SI, local SI_e, and λ over the iteration count.

See that, in every iteration, the worst SI_e among all scenarios is improved, which leads to an increase in the global SI. Note that the SI_e for the other scenarios may momentarily decrease, for example in $e = 3$ between the 2nd and 3rd iterations. However, this is needed to adjust

their solution to that of the worst scenario, which provides global convergence in the end.

Table 1. SI evolution over the iterations.

k	SI	SI_1	SI_2	SI_3	SI_4	λ
1	0.48	0.00	0.00	0.00	0.00	18.60
2	0.48	0.48	0.48	1.00	1.00	35.34
3	0.73	1.00	1.00	0.77	0.77	43.12
4	1.00	1.00	1.00	1.00	1.00	43.12

For the problem instance with 8 scenarios, the solutions and the cost functions were also identical, equal to 19927.4¤. The Gannt diagram is omitted for brevity. Here we observed a much more drastic improvement in the computation time. The decomposed approach (**47.35s**) was around 100 times faster than solving the monolithic problem (**4761.17s**). It shall be noted that the monolithic problem reaches a 2% optimality gap quickly, but struggles to find the optimal solution. We reckon that, by tuning the solver-specific heuristics, the performance of the monolithic formulation might be enhanced considerably, but it could also be the case for the subproblems in the decomposed approach.

5. CONCLUSIONS AND FUTURE WORK

This paper proposed a novel decomposition method for two-stage stochastic scheduling problems based on a single measure of the similarity between discrete-time schedules. The Similarity Index is computed by performing a fuzzification of the discrete decisions taken at some time over the neighboring instants. Then, an independent optimization problem is set up for every scenario, and the first-stage variables are forced to coincide using the Similarity Index.

The method was tested in two instances of an industrial-like problem with a different amount of scenarios. Such tests reported exponentially increasing savings in CPU time with the number of scenarios, with respect to a monolithic problem formulation. In other words, the proposed approach can exploit the advantages of parallel computation beyond the solvers native capabilities. Additionally, the subproblems could even be solved in different computers, in a grid-computing environment.

A disadvantage is that a feasible solution is only obtained when the problem has converged, i.e., the similarity index is equal to 1. In contrast, the monolithic approach looks first for a feasible solution and starts improving it. Furthermore, the convergence properties of the algorithm are not deeply analyzed yet. Thus, future work will focus on formal convergence analysis and on improving the multiplier update rule. We also aim to extend the method to continuous-time scheduling formulations and nonlinear stochastic scheduling problems.

ACKNOWLEDGEMENTS

These results are funded by the Spanish MICINN with FEDER funds, as part of the InCO4In (PGC2018-099312-B-C31) and LOCPU (PID2020-116585GB-I00) research projects. The first author has received financial support from the 2020 call of the pre-doctoral contracts of the University of Valladolid, co-financed by Banco Santander.

REFERENCES

Bazaraa, M.S. and Sherali, H.D. (1981). On the choice of step size in subgradient optimization. *Eur. J. Oper. Res.*, 7(4), 380–388.

Behrmann, G., Larsen, K.G., and Rasmussen, J.I. (2005). Optimal scheduling using priced timed automata. *ACM SIGMETRICS Perform. Eval. Rev.*, 32, 34–40.

Benders, J. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.*, 4, 238–252.

Crainic, G.T., Hewitt, M., Maggioni, F., and Rei, W. (2016). Partial Decomposition Strategies for Two-Stage Stochastic Integer Programs. *CIRRELT, Cent. Interuniv. Rech. sur les réseaux d'entreprise, la logistique le Transp. Interuniv. Rech. sur les réseaux d'entreprise, la logistique le Transp. .*, 37(July), 1–38.

Gade, D., Hackebeil, G., Ryan, S.M., Watson, J.P., Wets, R.J.B., and Woodruff, D.L. (2016). Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. *Math. Program.*, 157, 47–67.

Huang, Z. and Zheng, Q.P. (2020). A multistage stochastic programming approach for preventive maintenance scheduling of gencos with natural gas contract. *Eur. J. Oper. Res.*, 287, 1036–1051.

Palacin, C.G., Pitarch, J.L., de Prada, C., and Méndez, C.A. (2017). Robust multi-objective scheduling in an evaporation network. In *2017 25th Mediterranean Conference on Control and Automation (MED)*, 666–671.

Palacín, C.G., Pitarch, J.L., Jasch, C., Méndez, C.A., and de Prada, C. (2018). Robust integrated production-maintenance scheduling for an evaporation network. *Comput. Chem. Eng.*, 110, 140–151.

Palacín, C.G. (2020). *Efficient scheduling of batch processes in continuous processing lines*. Ph.D. thesis, Universidad de Valladolid.

Rahmaniani, R., Crainic, T.G., Gendreau, M., and Rei, W. (2017). The benders decomposition algorithm: A literature review. *Eur. J. Oper. Res.*, 259, 801–817.

Rockafellar, R.T. and Wets, R.J.B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.*, 16, 119–147.

Sahinidis, N.V. (2004). Optimization under uncertainty: state-of-the-art and opportunities. *Comput. Chem. Eng.*, 28, 971–983.

Sakawa, M. and Kubota, R. (2000). Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms. *Eur. J. Oper. Res.*, 120(2), 393–407.

Sawaya, N.W. and Grossmann, I. (2005). A cutting plane method for solving linear generalized disjunctive programming problems. *Comput. Chem. Eng.*, 29, 1891–1913.

Shor, N.Z. (1985). *The Subgradient Method*, 22–47. Springer Berlin Heidelberg, Berlin, Heidelberg.

Watson, J.P. and Woodruff, D.L. (2010). Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Comput. Manag. Sci.*, 8, 355–370.