Computers & Industrial Engineering 194 (2024) 110393

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Similarity-based decomposition algorithm for two-stage stochastic scheduling

Daniel Montes [a,c,*], José Luis Pitarch [b], César de Prada [a,c]

[a] Dpt. of Systems Engineering and Automatic Control, Universidad de Valladolid, C/Dr. Mergelina s/n, Valladolid, 47011, Spain
[b] Instituto U. de Automática e Informática Industrial (ai2), Universitat Politècnica de Valencia, Cno. de Vera s/n, Valencia, 46022, Spain
[c] Institute of Sustainable Processes (ISP), Universidad de Valladolid, C/Dr. Mergelina s/n, Valladolid, 47011, Spain

ARTICLE INFO

Keywords:
Production planning
Mathematical programming
Uncertainty
Progressive hedging
Mixed-integer optimization

ABSTRACT

This paper presents a novel decomposition method for two-stage stochastic mixed-integer optimization problems. The algorithm builds upon the idea of similarity between finite sample sets to measure how similar the first-stage decisions are among the uncertainty realization scenarios. Using such a Similarity Index, the non-anticipative constraints are removed from the problem formulation so that the original problem becomes block-separable on a scenario basis. Then, a term for maximizing the Similarity Index is included in all the sub-problems objective functions. Such sub-problems are solved iteratively in parallel so that their solutions are used to update the weighting parameter for maximizing the Similarity Index. The algorithm obtains a feasible solution when full similarity among scenario first stages is reached, that is, when the incumbent solution is non-anticipative. The proposal is tested in four instances of different sizes of an industrial-like scheduling problem. Comparison results show that the Similarity Index Decomposition provides significant speed-ups compared with the monolithic problem formulation, and provides simpler tuning and improved convergence over the Progressive Hedging Algorithm.

1. Introduction

Planning and scheduling deal with the optimal assignment of limited resources to processing tasks over time. Although deterministic scheduling formulations are mature, with broad industrial acceptance, they only translate to actual benefits in practice if the involved model parameters and inputs are precisely known. However, uncertainty over a future time horizon is inherent to all industrial processes. Some of the uncertain inputs that have important effects on the operations planning include: equipment up-time and availability, raw material arrival, prices, demands, weather, etc. Failing to correctly predict or estimate these values leads to suboptimal operation and can even result in infeasibility (Birge, 1995), where some operating constraints might need to be ignored to obtain a numerical solution.

A well-known approach to incorporating uncertainty into scheduling applications is multi-stage stochastic optimization. In particular, two-stage formulations are widely adopted due to their relevance to handle high-impact economic decisions. These formulations split the decision variables into two sets: first (here and now) and second (wait and see) stage variables and consider a set of scenarios with associated probabilities for the realization of the uncertain parameters. Unfortunately, these formulations increase the number of variables of the problem proportionally to the number of scenarios. This issue, combined with the fact that mixed-integer problems are already NP-hard in general, make two-stage formulations very challenging (or even impossible) to solve in reasonable amounts of time for online applications (Legrain, Omer, & Rosat, 2020; Palacín, Pitarch, Vilas, & de Prada, 2023; Sand & Engell, 2004; Simkoff & Baldea, 2020). Indeed, this strong limitation has received significant attention from the research community over the years: scenario bundling and reduction (Abouelrous, Gabor, & Zhang, 2022; Jiang, Bai, Wallace, Kendall, & Landa-Silva, 2021), reformulations, and decomposition methods (Ruszczyński, 2003) to split the monolithic problem into sub-problems that are easier to solve. Nevertheless, the problem still remains open and, regarding decomposition, the existing families of methods suffer from either high computational and memory demands (excessive useless Benders cuts (Lee, Ma, Yu, & Dai, 2021)) or tough parameter tuning to reach feasibility (multipliers update rules (Torres, Li, Apap, & Grossmann, 2022)).

We aim to contribute in this last way, by proposing a novel decomposition algorithm that does not build upon the few classical ideas of Benders, Lagrangean, or Progressive-Hedging decompositions. Here, we use a similarity indicator to split the monolithic problem into scenario-based sub-problems, whose solutions can be combined afterwards to

* Corresponding author.
E-mail address: danielalberto.montes.lopez@uva.es (D. Montes).

https://doi.org/10.1016/j.cie.2024.110393
Received 9 March 2023; Received in revised form 12 June 2024; Accepted 14 July 2024
Available online 17 July 2024

progressively obtain a feasible solution to the original problem while reducing significantly the overall computation time. Moreover, the problem size does not grow with iterations and the algorithm sensitivity to the few tuning parameters is not a critical factor for obtaining a feasible solution.

This paper extends the authors' preliminary idea in Montes, Pitarch, and de Prada (2022a) and Montes, Pitarch, and de Prada (2022b). While the base algorithm was roughly introduced in this conference material, a formal convergence and performance analysis still need to be addressed. Consequently, this paper provides proof of theoretical convergence and assesses tuning parameters. Furthermore, tuning parameters' effect on performance is verified through two-stage stochastic scheduling problems of varying sizes with binary first-stage variables compared to the progressive-hedging algorithm (PHA). The results support our claims of good performance in terms of convergence, optimality, and tuning easiness compared with the figures given by the PHA over the same benchmark.

The rest of the paper is organized as follows. Section 2 poses the problem statement and reviews the existing brands of solution methods in the literature. Section 3 explains how to use the Similarity Index (SI) to compare different scenario solutions (schedules). Section 4 describes the proposed decomposition algorithm based on such an index and analyzes its convergence properties. Section 5 presents the industrial-like case study that is used in Section 6 to test different aspects of the SI-decomposition method in comparison to the PHA. Finally, the paper closes with some concluding remarks and proposals for future work.

2. Problem statement

Let us consider the following two-stage stochastic scheduling linear formulation in discrete time and extensive form:

$$\underset{y,\, z_e,\, r_e}{\text{minimize}} \quad c^T y + \sum_{e\in\mathcal{E}} p_e q^T \begin{bmatrix} z_e \\ r_e \end{bmatrix} \tag{1a}$$

subject to

$$A y \geq b \tag{1b}$$

$$T_e y + W_e z_e + R_e r_e \geq h_e \qquad ,\forall e \in \mathcal{E} \tag{1c}$$

$$y := \{y_t \in \{0,1\}^{n_y} | t \in \mathcal{T}_1\} \tag{1d}$$

$$z_e := \{z_{te} \in \{0,1\}^{n_z} | t \in \mathcal{T}_2, e \in \mathcal{E}\} \tag{1e}$$

$$r_e := \{r_{te} \in \mathbb{R}^{n_s} | t \in \mathcal{T}_2, e \in \mathcal{E}\} \tag{1f}$$

The scenarios $e \in \mathcal{E}$ represent the future uncertainty realizations have and associated probabilities p_e, such that $\sum_e p_e = 1$. Time is discretized in periods $t \in \mathcal{T}$. The first stage is defined in the time periods $\mathcal{T}_1 \subset \mathcal{T}$ by the parameters in $A \in \mathbb{R}^{m_1 \times n_y}$ and $b \in \mathbb{R}^{m_1}$, weights $c \in \mathbb{R}^{n_y}$, and variables y_t. The second stage spans over the remaining time periods $\mathcal{T}_2 \subset \mathcal{T}$ and involves constraints per scenario defined by parameters $T_e \in \mathbb{R}^{m_2 \times n_y}$, $W_e \in \mathbb{R}^{m_2 \times n_z}$, $R_e \in \mathbb{R}^{m_2 \times n_s}$, $h_e \in \mathbb{R}^{m_2}$, weights $q \in \mathbb{R}^{n_z+n_s}$, as well as variables z_{te} and r_{te}.

To exploit the problem structure for decomposition purposes, the first-stage variables are disaggregated for each scenario so that they become y_{te} as in (2). However, for such a problem to be equivalent to (1), the so-called non-anticipativity constraints (2d) need to be enforced.

$$\underset{y_e,\, z_e,\, r_e}{\text{minimize}} \quad \sum_{e\in\mathcal{E}} p_e \left(c^T y_e + q^T \begin{bmatrix} z_e \\ r_e \end{bmatrix} \right) \tag{2a}$$

subject to

$$A y_e \geq b \qquad ,\forall e \in \mathcal{E} \tag{2b}$$

$$T_e y_e + W_e z_e + R_e r_e \geq h_e \qquad ,\forall e \in \mathcal{E}, \tag{2c}$$

$$y_{te_1} = y_{te_2} = \ldots = y_{t|\mathcal{E}|} \qquad ,\forall t \in \mathcal{T}_1 \tag{2d}$$

$$y_e := \{y_{te} \in \{0,1\}^{n_y} | t \in \mathcal{T}_1, \forall e \in \mathcal{E}\} \tag{2e}$$

$$z_e := \{z_{te} \in \{0,1\}^{n_z} | t \in \mathcal{T}_2, e \in \mathcal{E}\} \tag{2f}$$

$$r_e := \{r_{te} \in \mathbb{R}^{n_s} | t \in \mathcal{T}_2, e \in \mathcal{E}\} \tag{2g}$$

Note that if the non-anticipativity (2d) constraints were removed from the formulation, each scenario would represent an independent optimization problem. This is exactly what enables some of the decomposition methods available in the literature.

The Progressive Hedging Algorithm (Rockafellar & Wets, 1991) and Dual Decomposition in stochastic integer optimization (Carøe & Schultz, 1999) are the most widely used decomposition methods for two-stage scheduling problems and much of the current research focuses on building advanced algorithms upon them. Benders Decomposition (Benders, 1962) is also often used, but its main drawbacks are that the solution of the master problem becomes a limiting factor for large-scale problems (Mitrai & Daoutidis, 2022), and furthermore the presence of binary variables in the stage-2 problem produces a dual gap in the bound predicted by the master problem.

The Progressive Hedging Algorithm (PHA) combines the proximal method with the alternating direction method of the multipliers (Boyd, Parikh, Chu, Peleato, & Eckstein, 2010). It is often used for solving multi-stage stochastic optimization problems as it enables scenario-based decomposition by relaxing the non-anticipativity constraints. It is based on penalizing the deviation of the first-stage solution from an aggregated reference computed from the results obtained at a previous iteration. Although it was originally devised to handle convex problems involving only continuous variables, it is widely used as a heuristic for problems with integer variables (Khalilabadi, Zegordi, & Nikbakhsh, 2020; Peng, Zhang, Feng, Rong, & Su, 2019). Consequently, the convergence and optimality properties are not guaranteed for this last class of problems. Nonetheless, according to Gade et al. (2016) it is possible to assess the quality of the solutions by computing lower bounds at PHA iterations, and Watson and Woodruff (2011) proposed a set of improvements focusing on updating the penalty parameter, accelerating convergence, and enhancements to the termination criteria, albeit the main underlying limitations remain.

Dual Decomposition is a popular method for decomposing problems with continuous variables. Carøe and Schultz (1999) combined the Dual Decomposition method with a branch and bound algorithm to achieve convergence in two-stage and multi-stage stochastic problems. In this method, the non-anticipativity constraints are relaxed and a lower bound of the problem is obtained by solving the respective Lagrangean dual problem. This lower bound can be used to guide the computation of feasible solutions (upper bounds). The main difficulty associated with the Dual Decomposition method is that getting feasible solutions from problems without complete recourse is not guaranteed (Kim & Zavala, 2018). To overcome this limitation, Kim and Zavala (2018) proposed to add Benders-like cuts in order to tighten the subproblems and exclude infeasible first-stage solutions. The Dual Decomposition method has also been used together with the Benders Decomposition method in a hybrid approach to overcome the main drawbacks of the latter (Colonetti & Finardi, 2020). Although the literature is extensive, the success and performance of the available heuristic decomposition methods for mixed-integer programming is often problem dependent and strongly depends on finding a good parameter tuning (Torres et al., 2022).

3. Similarity index

The Jaccard, a.k.a. Tanimoto, coefficient (Gower, 1985; Tanimoto, 1958) is a widely used index in information and computer sciences to measure *similarity* among sets of binarized data (Willett, Barnard, & Downs, 1998). In brief, it is defined as the size of the intersection divided by the size of the union of the sample sets. Palacín, Pitarch, Jasch, Méndez, and de Prada (2018) used this concept in two-stage stochastic scheduling to obtain solutions up to a desired robustness

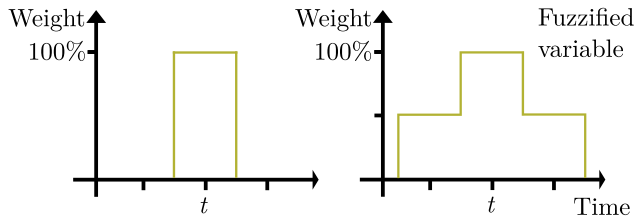


Fig. 1. Fuzzifying a decision along the following and previous time periods.

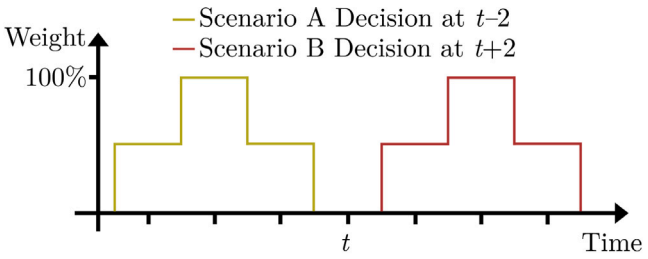
level. To accomplish such an aim, they proposed a Similarity Index (SI) as a way to quantify the risk of the computed stochastic schedule not being optimal (or even feasible) in practice, if the realized values of the uncertain inputs/parameters were not explicitly included within the scenario-tree approximation. In that context, the SI is a measurement of the similarity among *second-stage* decisions. Hence, a desired minimum robustness level can be enforced via lower bounding the SI in the mathematical MILP problem. In that case, specifying $SI = 100\%$ implies getting the *risk-averse* solution, whereas $SI \geq 0\%$ results in obtaining the less conservative two-stage stochastic solution.

Remark. Note that Palacín et al. (2018) did not use the SI to perform any problem decomposition. Their problem formulations remained monolithic after adding the uncertainty scenarios, which strongly limited the problem size that could be solved in a reasonable amount of time. Moreover, computing similarity among second-stage variables does not allow any kind of decomposition. In this work, the SI is used to force similarity among first-stage variables to progressively meet the non-anticipative constraints, as will be explained in Section 4. To the authors' knowledge, this is the first time the SI is used as the basis to build a decomposition algorithm for two-stage stochastic scheduling problems.

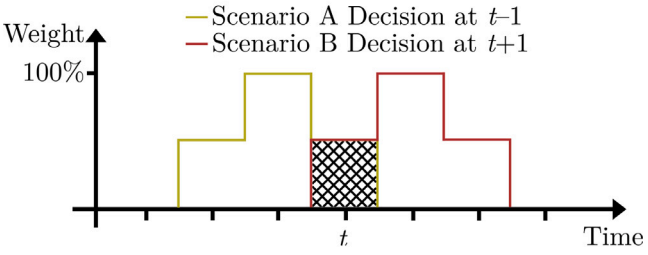
The decomposition method proposed in this paper to handle two-stage stochastic scheduling problems of the form (2) is based on the so-called Similarity Index. A *single* Similarity Index is used to replace all non-anticipativity constraints, naturally enabling scenario-based decomposition. The similarity between the scenarios solutions is then maximized in each iteration of the procedure until the first-stage solution is equal for all of the scenarios, that is, the first-stage solution is non-anticipative.

The core idea behind the SI is essentially the same as the Jaccard index, applied to compare discrete decisions among scenario schedules (binary decision sets) and weigh similarity if they are made in close, or neighboring, time periods. In a scheduling context, a given decision is said to be equal in two scenarios if it is made exactly at the same time period in both. Therefore, the Jaccard index could directly serve the task. However, what if, in one of those scenarios, such a decision is made in the previous or following time period? The schedules would not be the same, but they might be considered *similar*. Nevertheless, the Jaccard index weighs zero in such a case, because the intersection is null. An idea for providing the MIP optimizer with a better-conditioned problem and sensitivity is assigning positive weights $w \in \mathbb{R}$ (not depending upon scenarios) to each decision taken in the schedules to compare, and then compute the Jaccard index from the resulting non-binary sets (Willett et al., 1998). The proposed Similarity Index assigns such weights w by *fuzzifying* the discrete decisions along the surrounding time periods for each scenario as in Fig. 1. Hence, the contribution of a decision to the overall similarity decreases the further away it is made from the time period under comparison (that is, it is less similar).

Now, consider that there is a decision to be compared at time t among two uncertainty realization scenarios (A and B). However, the decision is made at $t-2$ and $t+2$ in scenario A and B respectively. The result of fuzzifying such decisions along two surrounding time periods



(a) No similarity at all.



(b) Fuzzified areas partially overlap.

Fig. 2. Examples of similarity computation when a particular decision is made for two different scenarios.

is shown in Fig. 2(a). It is clear that the fuzzified areas do not overlap at instant t so, consequently, the schedules are not similar at all. Then, $SI = 0$.

Fig. 2(b) shows the fuzzified areas generated by a decision made at $t \pm 1$ in scenarios A and B. The areas overlap partially in t, and one can say that the scheduled decisions are similar to a certain extent. The intersection area is 25% of the total area generated by the fuzzification. Then, $SI = 0.25$.

When the decisions are made at the same time instant there is complete overlap. Then, the generated intersection area at time t is the total one defined by the fuzzification. Thus, the schedules are 100% similar, that is, $SI = 1$.

The overall Similarity Index is conceptually defined as the total intersection area divided by the maximum intersection possible, i.e., the total area generated by the fuzzified decisions:

$$SI := \frac{\text{Intersection Area}}{\text{Maximum Area Possible}} \tag{3}$$

Hence, given a set of schedules corresponding to different scenarios, the overall SI is computed by aggregation of the intersected areas for each binary variable and time period in the schedule, divided by the total full-intersection area.

This concept can be applied to the first-stage variables to replace, or approximate, the non-anticipativity constraints. These constraints restrict the first-stage solution to be equal among the scenarios, which would imply that the Similarity Index is equal to 100%. Our decomposition method is based precisely on this idea. The non-anticipativity constraints are replaced by the Similarity Index, which is then iteratively pushed up through the objective function until it reaches 100%.

Definition. The overall Similarity Index for the first-stage discrete variables of a decomposed stochastic scheduling (2) is computed by:

$$SI := \frac{\sum_{t=1}^{t_R} \| \min_{e \in \mathcal{E}} \left\{ y_{te} + \sum_{\tau=1}^{\Delta} \frac{\Delta-\tau}{\Delta} (y_{(t-\tau)e}|_{t>\tau} + y_{(t+\tau)e}|_{t+\tau \leq t_R}) \right\} \|_1}{t_R \Delta - 2 \sum_{\tau=1}^{\Delta} \tau \frac{\Delta-\tau}{\Delta}} \tag{4}$$

where $y_{te} \in \{0,1\}$ are the binary decisions at time t in scenario e. $\Delta \in \mathbb{N}$ is the length of the fuzzification horizon in time periods. It includes

the current time period plus the ones to the right (or the left) which are considered for weighing similarity. $t_R \in \mathbb{N}$ is the number of time periods in the first stage, and t_1 is the first time period of this stage. $\tau \in \mathbb{N}$ is an auxiliary time index used to move along the periods in the fuzzification horizon, and weigh the decision made accordingly.

In this way, setting $\Delta = 2$ results in weighting only the immediate neighboring periods to the current one for similarity; if the two previous and following periods, besides the current one, are considered, then $\Delta = 3$, and so on. For instance, when $\Delta = 2$, (4) reduces to:

$$SI = \frac{1}{2t_R - 1} \sum_{t=1}^{t_R} \| \min_{e \in \mathcal{E}} \left\{ y_{te} + 0.5 y_{(t-1)e}|_{t>t_1} + 0.5 y_{(t+1)e}|_{t<t_R} \right\} \|_1 \tag{5}$$

Of course, the area of the extreme tails that would lie outside the robust horizon is not included in the total intersection area. Hence, $\Delta \leq \lceil t_R/2 \rceil$, as there is no sense in fuzzifying each decision outside the robust horizon.

Remark. Setting $\Delta = 1$ in (4) reduces to the standard Jaccard/Tanimoto index comparing binary data sets. Moreover, the denominator (i.e. the union between given sets in the Jaccard index) here is known beforehand and does not depend on the particular sets to compare, as any schedule is constrained to provide one and just one decision each time period. In this case, the union always coincides with the maximum intersection area. If a particular problem needs to allow a "no decision" possibility, extra decision variables need to be trivially added to account for not making any of the other decisions.

Note that the intersection operation, represented by the element-wise operator $\min\{\cdot\}$ in (4), is not linear. As will be stated later, the SI needs to be computed as part of the optimization problems, so this nonlinearity would become an issue for usual linear scheduling formulations.

Lemma 1. *The SI (4) can be lower bounded using the set of linear inequalities (6) and (7) together with slack variables $s \geq 0$.*

$$SI \geq \sum_{t \in \mathcal{T}_1} \frac{\|s_t\|_1}{t_R \Delta - 2 \sum_{\tau=1}^{\Delta} \tau \frac{\Delta - \tau}{\Delta}} \tag{6}$$

$$s_t \leq y_{te} + \sum_{\tau=1}^{\Delta} \frac{\Delta - \tau}{\Delta} \left(y_{(t-\tau)e}|_{t>\tau} + y_{(t+\tau)e}|_{t+\tau \leq t_R} \right) \forall e \in \mathcal{E}, \forall t \in \mathcal{T}_1 \tag{7}$$

Proof. On the one hand, by inequalities (7), s_t are strictly less than:

$$\min_{e \in \mathcal{E}} \left\{ y_{te} + \sum_{\tau=1}^{\Delta} \frac{\Delta - \tau}{\Delta} (y_{(t-\tau)e}|_{t>\tau} + y_{(t+\tau)e}|_{t+\tau \leq t_R}) \right\}$$

On the other hand, as the decisions y_{te} are binary and s_t are non-negative, the 1-norm in both (4) and (6) just performs the sum of its elements. Furthermore, s_t are always bounded by the maximum possible intersection area:

$$s_t \leq 1 + 2 \sum_{\tau=1}^{\Delta} \frac{\Delta - \tau}{\Delta}$$

Then, summing up for all $t \in \mathcal{T}_1$, excluding the tails that lie outside from the extreme periods, one obtains (6). □

For the above defined bound to be tight, the slack variables need to be added to the problem objective function for maximization.

3.1. SI computation example

Consider a two-stage scheduling problem with three discrete decision variables y_A, y_B, and y_C to be computed over five time periods (in the first stage). The uncertain parameters are discretized in three uncertainty realization scenarios. Table 1 shows a feasible schedule in which the first-stage decisions are equal among the scenarios (non-anticipativity is met).

Table 3 shows the contribution to the Similarity Index at each time period weighing only the following and previous time periods ($\Delta = 2$), that is, the sum of the fuzzified intersection areas of the decisions taken per scenario at each period. For instance, decision y_C is made at t_4 and t_5. There is a contribution of 0.5 at t_3 coming from the fuzzified decision made at the following period. At t_4, there is a contribution of 1 from the decision made at that period and 0.5 from the t_5 decision for a total contribution of 1.5. The same applies for t_5. Note that the total area is equal to 2 in the inner time periods, while in the extreme ones (t_1, t_5) it is equal to 1.5, as the tails of fuzzified areas would lie outside the first-stage horizon. In this case the intersection area among scenarios is always the maximum, as decisions are equal for all $e \in \mathcal{E}$.

Eq. (8) shows the computation of the SI for the schedule solution from Table 1 using (5).

$$SI = \frac{1.5 + 2 + 2 + 2 + 1.5}{2 \cdot 5 - 1} = \frac{9}{9} = 1 \tag{8}$$

As expected, the Similarity Index is equal to 100% for the schedule in Table 1. However, Table 2 shows an unfeasible schedule solution where some decisions are not the same among scenarios in the first stage.

As the decisions at t_3 are different, the Similarity Index must be less than 100%. Table 4 shows the intersection area at each time period, which is used in (9) to compute the SI. None of the decisions made at t_3 can contribute to the SI as they are non equal among the scenarios. The only contributions at t_3 come from the decisions made at t_2 and t_4. Note that the contribution to the SI is computed for each variable at each time period.

The unfeasible first-stage schedules from scenarios in Table 2 are then only 77.8% percent similar.

$$SI = \frac{1.5 + 1.5 + 1 + 1.5 + 1.5}{2 \cdot 5 - 1} = \frac{7}{9} = 0.778 \tag{9}$$

4. SI Decomposition algorithm

As mentioned in Section 1, removing the non-anticipativity constraints from two-stage stochastic scheduling problems allows decomposing the problem into smaller and easier-to-solve subproblems. However, the subproblem solutions would not coincide in the first-stage stage as nothing is enforcing non-anticipativity anymore. The core idea of our decomposition method is to solve the subproblems independently and use the Similarity Index to compare their solutions. An iterative procedure is set up to progressively increase the Similarity Index until complete similarity is reached, hence, meeting the non-anticipation criteria.

However, the SI formula shown in (4) is nonlinear and consequently cannot be used in the usual MILP formulations found in scheduling problems. Then, a local bound for the Similarity Index is computed for each subproblem using inequalities (6) and (7). Anyway, each subproblem requires information from the others to estimate the Similarity Index, and the decomposition would no longer be possible. The PHA and the Dual Decomposition solve this issue by using the solution obtained from each subproblem in the previous iteration. In our case, this is not possible as the inequalities (7) would bound the slack variables s_t to the worst contribution of all scenarios to the Similarity Index in each time period s_t. Eventually, the optimizer would be unable to increase the Similarity beyond a certain value. For instance, at time t_2 a scenario j could bound s_{t_2} to 0 as its solution is too different. Then, at time t_3, a scenario k could also bound s_{t_3} to zero. In the end, the optimizer would not be able to adjust the corresponding scenario solution to locally increase the similarity.

To solve this issue, our proposal is that subproblems only compare their decision variables (and thus measuring similarity) to a single reference set of decisions, denoted as $\bar{y}$. This reference is intended to be the solution of the subproblem in a previous iteration that is *less*

Table 1
Feasible first-stage schedule. All decisions are equal among scenarios, so non-anticipativity holds.

	t_1			t_2			t_3			t_4			t_5		
	e_1	e_2	e_3	e_1	e_2	e_3	e_1	e_2	e_3	e_1	e_2	e_3	e_1	e_2	e_3
y_A	1	1	1				1	1	1						
y_B				1	1	1									
y_C										1	1	1	1	1	1

Table 2
Unfeasible first-stage schedule. Decisions at t_3 are not equal in the three scenarios, so non-anticipation is not met.

	t_1			t_2			t_3			t_4			t_5		
	e_1	e_2	e_3	e_1	e_2	e_3	e_1	e_2	e_3	e_1	e_2	e_3	e_1	e_2	e_3
y_A	1	1	1				1		1						
y_B				1	1	1		1							
y_C										1	1	1	1	1	1

Table 3
Contribution of the feasible first-stage schedule in Table 1 to the SI.

	t_1	t_2	t_3	t_4	t_5
y_A	1	0.5 + 0.5	1	0.5	0
y_B	0.5	1	0.5	0	0
y_C	0	0	0.5	1 + 0.5	0.5 + 1
Total	1.5	2	2	2	1.5

Table 4
Contribution of the unfeasible first-stage schedule in Table 2 to the SI.

	t_1	t_2	t_3	t_4	t_5
y_A	1	0.5	0	0	0
y_B	0.5	1	0.5	0	0
y_C	0	0	0.5	1 + 0.5	0.5 + 1
Total	1.5	1.5	1	1.5	1.5

similar to the rest. With this in mind, the $|\mathcal{E}|$ subproblems derived from the monolithic problem (2), are formulated as in (10).

$$\underset{y_e, z_e, r_e, s}{\text{minimize}} \quad \mathrm{p}_e \left(\mathrm{c}^T y_e + \mathrm{q}^T \begin{bmatrix} z_e \\ r_e \end{bmatrix} \right) - \lambda \sum_{t \in \mathcal{T}_1} \frac{\|s_t\|_1}{t_R \Delta - 2 \sum_{\tau=1}^{\Delta} \tau \frac{\Delta - \tau}{\Delta}} \tag{10a}$$

subject to

$$\mathrm{A} y_e \geq \mathrm{b} \tag{10b}$$

$$\mathrm{T}_e y_e + \mathrm{W}_e z_e + \mathrm{R}_e r_e \geq \mathrm{h}_e \tag{10c}$$

$$s_t \leq \bar{y}_t + \sum_{\tau=1}^{\Delta} \frac{\Delta - \tau}{\Delta} \left(\bar{y}_{(t-\tau)}|_{t>\tau} + \bar{y}_{(t+\tau)}|_{t+\tau \leq t_R} \right) \quad , \forall t \in \mathcal{T}_1 \tag{10d}$$

$$s_t \leq y_{te} + \sum_{\tau=1}^{\Delta} \frac{\Delta - \tau}{\Delta} \left(y_{(t-\tau)e}|_{t>\tau} + y_{(t+\tau)e}|_{t+\tau \leq t_R} \right) \quad , \forall t \in \mathcal{T}_1 \tag{10e}$$

$$y_e := \{ y_{te} \in \{0,1\}^{n_y} | t \in \mathcal{T}_1 \} \tag{10f}$$

$$z_e := \{ z_{te} \in \{0,1\}^{n_z} | t \in \mathcal{T}_2 \} \tag{10g}$$

$$r_e := \{ r_{te} \in \mathbb{R}^{n_s} | t \in \mathcal{T}_2 \} \tag{10h}$$

$$s := \{ s_t \in \mathbb{R}^{n_y} | t \in \mathcal{T}_1 \} \tag{10i}$$

The reference values $\bar{y}_t \in \{0,1\}^{n_y}$ that appear in (10d) refer to the solution in the previous iteration of the scenario sub-problem that yielded the worst *local* Similarity Index. The parameter λ weights the importance of maximizing such index: a value close to zero would mean that the original objective function is prioritized over maximizing the similarity; however, if the Similarity Index is not maximized, it is highly probable that the obtained solutions do not meet the non-anticipation criteria in the end. Contrarily, a large value of λ would give more importance to maximizing the similarity, at the cost of getting to a worse solution in the terms of optimality. Therefore, inspired by the sub-gradient method (Shor, 1985), we propose to update the weighing parameter λ as in (11).

$$\lambda_{k+1} = \lambda_k - \alpha \left(SI_k - 1 \right) \tag{11}$$

where k represents the current iteration, α is a tuning parameter that represents the step size, and SI_k is the exact *global* Similarity Index computed a posteriori using (4) in each iteration from the subproblems solutions.

Algorithm 1 provides the pseudo-code of the decomposition method. In the algorithm, J_e stands for the cost of scenario e, that also represents the contribution of each scenario solution to the overall cost according to the probability distribution, i.e.:

$$J_e := \mathrm{p}_e \left(\mathrm{c}^T y_e + \mathrm{q}^T \begin{bmatrix} z_e \\ r_e \end{bmatrix} \right) \tag{12}$$

Algorithm 1 Similarity Index Decomposition

Require: $\alpha, \mathrm{k}_{\max}, \Delta$
1: $k \leftarrow 0, \lambda_0 \leftarrow 0, \bar{e} \leftarrow \emptyset$ ▷ Initialization
2: $\bar{y} \leftarrow \arg\min_y c^T y$, s.t.: $t \in \mathcal{T}_1$ ▷ Solve first-stage deterministic problem
3: **repeat**
4: **parallel for** e in $\mathcal{E}/\bar{e}$ **do**
5: $y_e^*, z_e^*, r_e^* \leftarrow \arg\min_{y_e, z_e, r_e, s} J_e - \lambda_k SI_e$ ▷ Solve (10)
6: **end parallel for**
7: $\ell \leftarrow \arg\min_{e \in \mathcal{E}/\ell} SI_e(y_e^*, \bar{y})$ ▷ Scenario(s) with worst local SI
8: $\bar{e} \leftarrow \arg\max_{e \in \ell} J_e(y_e^*, z_e^*, r_e^*)$ ▷ Scen. with higher cost contribution
9: $\bar{y} \leftarrow y_{\bar{e}}^*$
10: $\mathrm{SI} \leftarrow SI(y_{e \in \mathcal{E}}^*)$ ▷ Global SI computation
11: $\lambda_{k+1} \leftarrow \lambda_k - \alpha \cdot (\mathrm{SI} - 1)$
12: $k \leftarrow k + 1$
13: **until** $\mathrm{SI} = 1 \vee k = \mathrm{k}_{\max}$ ▷ Convergence check
14: **if** SI=1 **then return** $y_{te}^*, z_{te}^*, r_{te}^*$
15: **else** No feasible solution found
16: **end if**

The algorithm begins by initializing the weighing parameter λ and the iteration counter k to zero, and an empty set $\bar{e}$. Then, the reference values $\bar{y}$ are initially set to the solution of the deterministic problem consisting in formulation (1) restricted to the first stage, i.e., just considering variables and constraints for $t \in \mathcal{T}_1$. Note that this is a fast-to-obtain initial guess required to compute local similarity indexes SI_e in the first iteration. Then, the subproblems (10) can be solved in parallel and their solutions are collected (line 4). Afterwards (line 7), the algorithm seeks for the subset of scenarios $\ell \in \mathcal{E}$ that showed the lowest SI_e value (computed a posteriori by (4) with just the solution y_e^* of each subproblem and the reference $\bar{y}$). Subsequently, the scenario from those in ℓ which is contributing most to the overall cost function is selected (line 8) and used to set the new reference $\bar{y}$

(line 9). The global SI is computed (line 7) and used to update the weighing parameter λ (line 9) by rule (11). The process repeats until full similarity (convergence) or the maximum number of iterations k_{max} are reached (line 13). If convergence is reached, the algorithm output is the first-stage and recourse decisions.

Remark. Note that, in any iteration $k > 0$, the scenario subproblem $\bar{e}$ corresponding to the current reference $\bar{y}$ does not need to be solved in the next iteration. This is because $\lambda_{k+1} > \lambda_k$ so, evidently, the optimal solution for this subproblem is again $y^*_{\bar{e}} = \bar{y}$ that makes $SI_{\bar{e}} = 1$. This is of utmost importance with limited CPU resources, as it frees some threads/cores to be directly assigned to other subproblems. Note also that for $k > 0$, the computations benefit from the native warm-start feature of the MILP solvers and extra time is saved by initializing subproblems at their previous solutions.

4.1. Convergence & optimality

The procedure that Algorithm 1 describes seeks to progressively push the scenario different solutions together to reach 100% similarity in the first stage. In brief, the procedure uses the local SI_e to select the scenario whose solution is less similar to the reference one, and so on. Informally, this is, the first-stage decisions for the "reference scenario" are frozen and the other scenario solutions are pushed closer to them. Then, at next iteration, such reference is changed to the scenario whose solution in the first stage remained further away from the reference. In case there are multiple scenario solutions at "equally far distance", the one that contributes most to the overall cost function is set as an heuristic for the reference choice. Note that this concept of distance between two discrete-decision schedules (a measure of how "far apart" they are) can be formally stated by its *symmetric difference*, a.k.a. disjunctive union (Flament, 1963).

Lemma 2. *Given two vectors y_1 and y_2 of dimension n over the binary numbers and the Similarity Index defined in (4), its symmetric difference*

$$\delta(y_1, y_2) := 1 - SI(y_1, y_2) \tag{13}$$

is a proper measure of the distance between them.

Proof. The SI is a real-valued function $\Psi : \{0,1\}^n \to \mathbb{R}_+$ upper bounded by 1 by definition, so it is δ too. In addition, the following properties hold:

1. $\delta(y, y) = 0 \ \forall y \in \{0,1\}^n$
2. $\delta(y_1, y_2) = \delta(y_2, y_1) \ \forall y_1, y_2 \in \{0,1\}^n$
3. $\delta(y_1, y_3) \leq \delta(y_1, y_2) + \delta(y_2, y_3) \ \forall y_1, y_2, y_3 \in \{0,1\}^n$

Property 3 (triangle inequality) is proven from the property of the inverses in a Boolean group (Rudin et al., 1976). □

Hence, $\delta = 0$ means full similarity whilst $\delta = 1$ means no similarity at all.

Proposition 1. *Assume that a feasible solution for problem (1), equivalently (2), exists. Then:*

1. *The solution of (10) with $\lambda = 0$ also exists for each scenario $e \in \mathcal{E}$. Thus, cost (2a) computed from such solutions states a lower bound on the optimal cost (1a).*
2. *Any solution set obtained from independent sub-problems (10) with $\lambda > 0$, fulfilling $SI = 1$ is an upper bound on the optimal cost (1a).*

Proof. Problem (10) stated for each scenario with $\lambda = 0$ is equivalent to (2) without non-anticipativity constraints (2d). That is, a relaxation of (1), which proves statement 1. Statement 2 is evident, as any solution set fulfilling all constraints in (2) is a feasible solution for (1). □

Theorem. *If a feasible solution for (2) exists, Algorithm 1 with $\alpha > 0$ reaches it in a finite number of iterations. Moreover, with α large enough it reaches a feasible solution in $|\mathcal{E}| + 1$ iterations at most.*

Proof. Consider the set of solutions $y^*_{e\in\mathcal{E}}$ provided by Algorithm 1 at any iteration k. Denote by Θ to the sum of all peer-to-peer distances between solutions y^*_e. Let $\bar{y} = y^*_{\bar{e}}$ be the solution of the selected reference scenario $\bar{e}$. Then,

$$\Theta := \sum_{i=1}^{|\mathcal{E}|}\sum_{j>i}^{|\mathcal{E}|} \delta(y^*_i, y^*_j) = \sum_{\substack{i,j\in\mathcal{E}/\bar{e}\\ j>i}} \delta(y^*_i, y^*_j) + \sum_{e=1}^{|\mathcal{E}|} \delta(y^*_e, \bar{y})$$

$$\leq \sum_{\substack{i,j\in\mathcal{E}/\bar{e}\\ j>i}} \Big(\delta(y^*_i, \bar{y}) + \delta(y^*_j, \bar{y})\Big) + \sum_{e=1}^{|\mathcal{E}|} \delta(y^*_e, \bar{y})$$

by $\delta(\cdot,\cdot)$ satisfying the triangle inequality (Lemma 2). Algorithm 1 progressively maximizes each $SI_e(y^*_e, \bar{y})$ through λ_k, thus minimizing $\delta(y^*_e, \bar{y}) \forall e \in \mathcal{E}$ and, in particular, the largest distance by setting $\bar{y}$ to the scenario with worst local SI. Furthermore, as λ_k increases monotonically with k, Algorithm 1 performs a contractive process. Consequently, $\Theta_k \leq \Theta_{k-1}$ holds. Then, two situations can occur: either Θ converges to a positive value or to zero. Θ converging to a value different from zero would mean that there is no feasible solution, which contradicts the main assumption of the theorem. Therefore, if a feasible solution for (2) exists, eventually Θ must reach zero which, in turn, means the global SI = 1 as $\Theta \leq |\mathcal{E}|\big(1 - SI(y^*_{e\in\mathcal{E}})\big)$ by definition (recall the min operator in (4)).

Now, let $\alpha \gg 0$ be large enough such that $\lambda_k \geq \lambda^* \ \forall k$, being λ^* the optimal value[1] of the Lagrange multiplier in problem (10)[2] to fulfill:

$$\sum_{t\in\mathcal{T}_1} \frac{\|s_t\|_1}{t_R\Delta - 2\sum_{\tau=1}^{\Delta} \tau\frac{\Delta-\tau}{\Delta}} = 1$$

In this case, problem (10) focuses on fulfilling the equality above. All solutions $y_{e\in\mathcal{E}}$ are forced to fully meet the reference $\bar{y}$ (Lemma 1) from $k > 0$. In this setup, if any $y^*_{\mathbf{e}} \neq \bar{y}$ after solving the sub-problems, it is because $\bar{y}$ is infeasible for that sub-problem $\mathbf{e}$. Consequently, Algorithm 1 sets $\bar{y} = y^*_{\mathbf{e}}$ for $k + 1$ and forces the remaining $y_{e\in\mathcal{E}}$ to meet that reference. In the worst case, Algorithm 1 has tested all vertices $y^*_{e\in\mathcal{E}}$ as reference in $k = |\mathcal{E}| + 1$ iterations, so a feasible solution must have been reached. □

Of course, reaching a global feasible solution does not guarantee optimality, as any feasible solution different from the optimal is just an upper bound, see Proposition 1.

Corollary. *The optimal solution for (2) lies in the convex hull formed by the $|\mathcal{E}|$ vertices of optimal solutions $y^*_{e\in\mathcal{E}}$ computed from (10) with $\lambda = 0$.*

Proof. These scenario optimal solutions $y^*_{e\in\mathcal{E}}$ define a lower bound on the cost (Proposition 1). Indeed, this bound is the lowest possible to be computed from independent scenario solutions, otherwise would mean that the solution for any scenario is suboptimal. This also means that there cannot exist a different set of solutions $y_{e\in\mathcal{E}}$ yielding better overall cost with higher peer-to-peer distance sum Θ. Consequently, as (2) is a linear programming problem, its optimal solution y^* must lie in the convex hull defined by $y^*_{e\in\mathcal{E}}$ obtained with $\lambda = 0$. □

[1] Karush-Kuhn–Tucker (KKT) necessary optimality conditions.

[2] The duality statement and KKT conditions are extended to problems where some of the variables are integer and the objective and constraints are convex (Baes, Oertel, & Weismantel, 2016).

4.2. Parameter tuning

Algorithm 1 has mainly two tunable parameters: the step size $\alpha \in \mathbb{R}$ and the length of the fuzzification horizon $\Delta \in \{2, 3, 4, \dots\}$. Intuitively, step size α balances solution optimality with convergence speed: a large value of α implies that more importance is given to maximizing the Similarity Index at the cost of possibly reaching a suboptimal solution. On the contrary, small α provides more flexibility for the subproblems to progressively adapt their schedules towards the feasible optimum, but many iterations might be required to converge.

The length of the fuzzification horizon Δ affects the granularity of the SI function. Although $SI \in \mathbb{R}$, by definition, it can only take a finite number of values. Thus, larger Δ increases the number of possible values the SI can take. There is no clear insight on what is the influence of this parameter in the solution quality for any general case. On the one hand, large Δ implies that, for each decision that is not coincident in a given period t among the scenarios, more neighboring periods get their SI reduced, thus reducing the overall SI value. This means that finding the right path to improve the SI at first iterations is harder for the optimizer, because the SI sensitivity is lower. On the other hand, the smallest Δ may affect SI sensitivity in a similar way if a decision is scheduled very close in time among scenarios but just outside the fuzzification horizon. For instance, if certain constraints in a scenario impede to take as first decision (at t_1) the one that is considered optimal for the rest of scenarios (e.g. a maintenance task), and then it is taken with a delay of just Δ periods, the schedules could be exactly the same but slightly delayed in time. Clearly some similarity exists in this hypothetical case. However, the overall SI will be zero as only the immediate neighboring period is considered towards similarity.

From this analysis one can infer that the problem definition itself is which could determine the optimal length of the fuzzification horizon: same decision taken with a difference of Δ periods may be seen as relatively similar if the scheduling horizon (more precisely, $|\mathcal{T}_1|$) is large, or dissimilar if the opposite. Hence, for problems where the first stage involves many periods, Δ shall be increased from the default value 2 (by which only periods $t-1$ and $t+1$ are weighted towards similarity) to keep a desired ratio $\Delta/|\mathcal{T}_1|$.

5. Case study

The performance of the Similarity Index Decomposition is analyzed with the industrial case studied by Palacín et al. (2018) under the umbrella of a European Research and Innovation Action (CoPro, 2020). In such a work, a discrete-time MILP formulation was proposed to schedule the production and maintenance tasks in an evaporation network of a cellulose fibers manufacturing plant. In this process, some chemicals (henceforth products) are used in the spinning process to provide the fibers with some desired mechanical properties. These products have a lower concentration after going through the main process, so they are sent to a network of evaporators to be concentrated. Each of these evaporation plants consists of a series of heat exchangers, evaporation chambers, condensers, and cooling systems. A surrogate model was built to estimate the operating costs of each evaporation plant. This cost model depends on the efficiency of its cooling system, the nominal efficiency of the evaporation plant, the increased costs due to fouling, and the product load.

The goal of the optimization problem is to minimize the operating costs of the evaporation network over a finite horizon denoted by discrete time periods $t \in \mathcal{T}$. The evaporation plants $v \in \mathcal{V}$ can only process a product $p \in \mathcal{P}$ at a time. Unfortunately, the plants need to be regularly cleaned as fouling decreases their efficiency increasing steam consumption and thus the operating costs. The continuous fouling effect is discretized over a set of consecutive states $\{s_1, \dots, s_{40}\} \in \mathcal{S}$ with decreasing plant efficiency. The optimizer needs to decide when to stop the plants to perform either a deep (A) or a light (B) cleaning, which are denoted by states $s_A, s_B \subseteq \mathcal{S}$. The plants can also be put on stand-by either before or after a cleaning task. Time is discretized in one-day-long periods as that is what usually cleaning the plants takes.

The optimization problem seeks to optimally schedule the evaporation plants and their maintenance (cleaning). Production is modeled by non-negative real variables P_{vtp}. Product assignment and plant states are decided using binary variables A_{vtp} and E_{vts} respectively.

The model accounts for several constraints including:

- Evaporation plants can only be in one and only one state at each time period.
- Evaporation plants can only process one product at a time.
- Due to personnel limitations, only one cleaning task can occur per time period.
- After an evaporation plant is cleaned, it can only be cleaned again after several time periods in operation.
- An evaporation plant in operation can stay in such a state, go to stand-by, or get cleaned. A fouled plant on standby can only remain on standby or be cleaned. A clean plant can remain on stand-by or start operating.
- The evaporation plants cannot change products unless a cleaning task is performed.

Uncertainty is very important for the adequate operation of the network. The product demand D_p can vary over the prediction horizon depending on the market and the main process. Additionally, the maximum load of the evaporation plants depends on the ambient air temperature. The model is then formulated as a two-stage stochastic scheduling problem with scenarios $e \in \mathcal{E}$. This additional index e is added to the decision variables, together with non-anticipation constraints. For a detailed description of the model and its equations, refer to Palacín et al. (2018).

The SI formula needs to be updated to the case study nomenclature and additional sets. The SI will only be calculated with respect to the evaporation plant state variables E_{vtse} (binary). Due to the problem's particular structure, fixing E_{vtse} implies unique values for product allocation (A_{vtpe}) and plant load (P_{vtpe}) decisions.[3] Of course, this would not be the case in general, but the reader may note that the SI-decomposition algorithm can trivially combine with PHA for instance, to handle continuous variables appearing in the first stage. Details omitted for brevity.

Hence, the formula to compute the overall SI in this case, shown in (14), only accounts for E_{vtse}. The corresponding formula for the local SI (SI_e) is omitted, as it can be easily derived from (14).

$$SI = \sum_{v\in\mathcal{V}} \sum_{s\in S} \sum_{t\in\mathcal{T}_1} \frac{\| \min_{e\in\mathcal{E}} \left\{ E_{vtse} + 0.5E_{v(t-1)se} + 0.5E_{v(t+1)se} \right\} \|_1}{n_v \left(2t_R - 1\right)} \tag{14}$$

The main apparent difference of (14) with respect to the formula (5) in the SI definition is that there are two extra summations according to the sets defining the binary variables: the evaporation plants $\mathcal{V}$ and the operation state S. Note that the original denominator is multiplied by the number of evaporators n_v, but not by the number of states, as the evaporators can only be at one state at a time.

The PHA was used to assess the performance of the SI decomposition. Our implementation of the PHA is shown in Algorithm 2. Note that no update rule was performed for the parameter ρ, so it remains constant for every iteration.

6. Benchmark results & discussion

This section presents the performance assessment among the Similarity Index Decomposition, the Progressive Hedging Algorithm, and

[3] Plants current state (at t_0) is not a decision variable. So, once E_{vtse} is fixed equally for all $e \in \mathcal{E}$, constraints on product changeover and cleaning prevent the existence of different allocations among scenarios.

Algorithm 2 Progressive Hedging Algorithm

Require: $\rho, \mathrm{k}_{\max}$
1: $k \leftarrow 0, \lambda^0_{vste} \leftarrow 0$ ▷ Initialization
2: **for** e in $\mathcal{E}$ **do** ▷ Decomposition
3: $E^*_{vste} \leftarrow \mathrm{argmin}_{E_{vste}} J_e$
4: **end for**
5: **repeat**
6: $k \leftarrow k + 1$
7: $\widehat{E}_{vst_R} = \frac{1}{|\mathcal{E}|} \sum_{e \in \mathcal{E}} E^*_{vst_R e}$ ▷ Aggregation
8: $\lambda^k_{vst_R e} \leftarrow \lambda^{k-1}_{vst_R e} + \rho \left(E_{vst_R e} - \widehat{E}_{vst_R} \right)$ ▷ Price update
9: **for** e in $\mathcal{E}$ **do** ▷ Decomposition
10: $E^*_{vste} \leftarrow \mathrm{argmin}_{E_{vste}} J_e + \lambda^k_{vst_R e} \cdot E_{vst_R e} + \frac{1}{2}\rho \| E_{vst_R e} - \widehat{E}_{vst_R} \|^2$
11: **end for**
12: **until** $\| E^*_{vst_R e} - \widehat{E}_{vst_R} \| = 0 \vee k = \mathrm{k}_{\max}$ ▷ Stop criteria
13: **return** E^*_{vste}

the Monolithic formulation, in solving four instances of the above case study but of increasing size. The performance criteria chosen to assess the algorithms are solution quality, computation speed, and sensitivity to the tuning parameters. The four problem instances are defined by:

- Instance A: 3 plants, 2 products, 4 scenarios and 25 days horizon. 21 869 variables.
- Instance B: 9 plants, 3 products, 16 scenarios and 25 days horizon. 187 202 variables.
- Instance C: 9 plants, 3 products, 16 scenarios and 30 days horizon. 224 642 variables.
- Instance D: 23 plants, 5 products, 16 scenarios and 30 days horizon. 618 242 variables.

For the sake of comparison, the robust horizon $\mathcal{T}_1$ was set to 7 days for all the instances. It shall be noted that approximately 90% of the variables are binary. The modeling environment GAMS 42.1.0 together with Gurobi 10.0 as MIP solver were used for implementation. The monolithic and SI decomposition instances resulted in MILPs, while the PHA ones lead to MIQPs.

6.1. Efficiency and effectiveness

Table 5 presents a comparison of the cost objective value and computation time of the different approaches in all the case study instances. The stop criterion for the optimizer Gurobi was the optimality gap[4] in the decomposition approaches whilst a time limit of 10 days of computation is set for the monolithic approach. The step sizes ρ, α, for the PHA and SI decomposition respectively, were tuned to obtain the minimum cost (objective function) at the lowest possible computation time. $\Delta = 2$ was set for the SI decomposition.

In Instance A, both decomposition methods were considerably slower than their monolithic counterpart. The reported objective values were identical. However, this instance is apparently too small to truly realize the benefits of decomposition. Instance B showed significant time reductions for both decomposition approaches, two orders of magnitude faster than the monolithic approach. Note that this is even stopping the time count in the monolithic resolution at 0.49% gap, when the finally proven (after 4412 s) optimal value was first found. In this instance, while SI Decomposition reached the optimal cost value, the PHA reported a slightly worse cost. For instance C, both decomposition approaches greatly outperformed the monolithic approach in terms of computation time, again despite of stopping the clock of the monolithic resolution when the optimal solution is first found, almost at 1% gap (to reduce such gap to 0, Gurobi spent 176.5 h in total). In this case, PHA slightly outperformed SI Decomposition for the chosen setups. Finally, in the largest instance D, the SI Decomposition only needed 4 iterations to converge to a solution whose cost is better than the one obtained by PHA after comparable computation time, and much better than what the monolithic approach is able to provide after ten days.

For completeness, Fig. 3 graphically shows the progress of Algorithm 1 with the iterations. Note that, although some distances δ can

[4] Gap between the incumbent solution and the theoretically best possible as for the current branch-and-bound exploration.

Table 5
Comparison on the four problem instances solved by the monolithic approach, the PH algorithm, and the SI decomposition.

Instance	Approach	Step size	Cost	Time (s)	Iter.	Gap (%)
A	Monolithic	–	10 051.6	8	–	0
	PHA	2.5	10 051.6	27	5	0
	SI-D	460	10 051.6	24	3	0
B	Monolithic	–	91 819.6	1305	–	0.49
	PHA	45	91 843.05	37	4	0
	SI-D	700	91 819.6	40	3	0
C	Monolithic	–	108 698.4	48 569	–	0.97
	PHA	45	108 698.4	417	4	0
	SI-D	1500	108 730.4	565	3	0
D	Monolithic	–	527 617	864 000	–	> 1
	PHA	45	527 298	12 737	6	0.01
	SI-D	2500	527 037	15 470	4	0.01

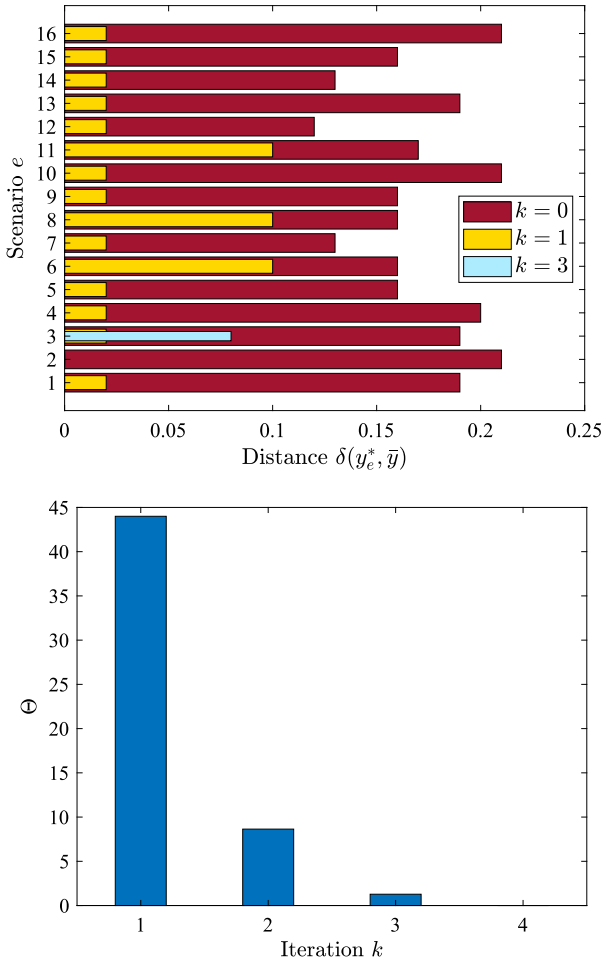


Fig. 3. Computed distances through the optimization process of Instance D with SI-D. Solutions y^*_e = 2, 8 and 3 were chosen as reference in iterations k = 1, 2 and 3, respectively.

Table 6
Effect of Δ on the SI decomposition algorithm.

Instance	Δ	Cost	Iter.	Time (s)
A	1	10 051.6	3	28
	2	10 051.6	3	24
	3	10 051.6	3	27
	4	10 051.6	3	28
B	1	91 819.6	3	38
	2	91 819.6	3	40
	3	91 819.6	3	45
	4	91 819.6	3	43
C	1	108 698.4	4	630
	2	108 730.4	3	565
	3	108 730.4	3	726
	4	108 730.4	3	620
D	1	527 375.6	3	13 347
	2	527 037.1	4	15 470
	3	527 375.3	5	21 332
	4	527 375.3	5	21 660

eventually increase after switching the reference solution, Θ always decreases towards zero with the iterations.

All tests and parallel computations were performed on a Dell Precision T5500 workstation with two Intel Xeon X5650 CPUs (24 available logical processors in total), setting three threads in Gurobi to solve each subproblem.

6.2. Effect of the fuzzification horizon

As discussed in Section 4.2, the length of the fuzzification horizon (in which similarity is considered among variables in different scenarios) is determined by the parameter Δ. Hence, $\Delta = 2$ means that similarity is considered only within the previous and following period. For $\Delta = 3$, two previous and two following days are taken into account, and so on.

To test the influence of Δ in the SI Decomposition, all problem instances were solved for $\Delta \in \{1, 2, 3, 4\}$ so that the number of surrounding days to weight in the SI is between 2 and 7. The objective values and computation times are shown in Table 6.

The parameter Δ did not significantly impact the cost value and computation times in the small problem instances. However, some differences can be observed both in optimality and computation time for the larger instances, especially instance D. On instance C, $\Delta = 2$ provided the shortest computation time by approximately 10%, but yielded higher cost than with $\Delta = 1$. Effects look more appreciable on instance D, where computation times monotonically increase with Δ, but note that the best cost value is obtained with $\Delta = 2$.

These numerical tests seem to confirm that the effect of Δ on the decomposition method is minor and might depend on the problem structure and size. As a tuning guideline, the modeler may set Δ to low values and only increase it to explore if there is an extra improvement to unlock in problems where $|\mathcal{T}_1|$ is large, i.e., fine discretized robust horizons.

6.3. Effect of the step size in the SI decomposition

Instances A and D were solved for different values of α. Fig. 4 shows the computation time and the iterations required to converge in such instances for the tested values. For this evaluation, the problem objective functions were not scaled in neither case.

The algorithm converged regardless of the value of α, as expected from theory. Small values of α lead to many iterations and, consequently, to larger computation times. For large values of α, although the computation time was shorter in both instances, the resulting solutions deviated slightly from optimality: the gap became 0.2% for $\alpha >= 500$ in Instance A, whilst an increase of 0.4% for $\alpha >= 5000$ was reported in Instance D. Values in the ranges of $100 < \alpha < 500$ in Instance A and of $1000 < \alpha < 5000$ in Instance D are the ones which reported the best performance if optimality is prioritized. For larger α values the computation times continue decreasing, but tend to converge to the minimum number of iterations possible. Consequently, the impact of varying α within relatively large values is not significant.

As a suggestion, the algorithm shall be tuned starting with a large value of α and then decrease it until the computation time increases significantly (i.e. exceeds what is considered acceptable for a particular application). Then, the value for which the best objective is obtained at the lowest CPU time would be the choice. The opposite path is not recommended, as the algorithm would need many iterations to converge in the first trials.

6.4. Effect of the step size in the PHA

Although the step sizes of the PHA and our algorithm are not directly comparable because of their different meaning in the formulations, the performance of the PHA was evaluated as well using different values of ρ. Fig. 5 shows the computation time and the required number of iterations in Instance A for different values of ρ.

The PHA converged to the optimal solution for $\rho \in [0.1, 3)$. As seen in figure, the computation time tends to decrease as ρ increases, until reaching a limit value where the algorithm did not converge to any feasible solution. Outside the provided interval, the error criterion $\|E^*_{vst_R e} - \hat{E}_{vst_R}\|$ gets stuck in a positive value after some iterations. This issue was amplified in instances B, C and D where, despite the different problem sizes and initial conditions, convergence was only achieved in a small range around $\rho = 45$.

An explanation for this unwanted behavior is in the discontinuity of the feasible region due to binary decision variables (Rockafellar & Wets, 1991). What probably happens in this case is that, when a few scenario optimal solutions diverge from the rest and cannot approach the reference (mean) values $\hat{E}_{vst_R}$ for some reason (either a specific constraint or because multiplier λ already became too large), all scenario solutions keep invariant. And this can happen for too large or too small values of ρ, as scheduling problems often involve numerous lower cost operational decisions. Besides, oscillation phenomena can also occur if λ grows too fast (Watson & Woodruff, 2011). Consequently, in a complex setting, tuning ρ is a non-directed trial-and-error process for finding a value that just gets convergence, becoming tougher if best performance-optimality tradeoff is sought.

7. Conclusions

This work shows how to use a similarity measure among discrete data sets to replace the non-anticipativity constraints in two-stage stochastic scheduling problems. This allows decomposing the monolithic problem into smaller subproblems on a scenario basis. The proposed method significantly reduced the required computation time to obtain a solution in realistic-size instances of the presented case study, without sacrificing the actual optimization objective. The results gathered indicate that such speed-up would enable considering uncertainty in near real-time plant scheduling: note that the largest instance presented is similar in size to the actual industrial facility the work is based on, and it would have been impossible to solve in sensible time without a decomposition approach. Moreover, in about four hours of computation, the SI Decomposition provided a higher quality solution than the one reached by the monolithic formulation in ten days of computation.

Compared to the existing alternatives in the literature, such as the Progressive Hedging Algorithm, this new method is advantageous in the sense that the formulation is more compact (non-anticipativity in all first-stage variables is merged into a single indicator) and simpler to tune: the modeler can balance between optimality and convergence speed by just varying the step size α. In contrast, for the PHA

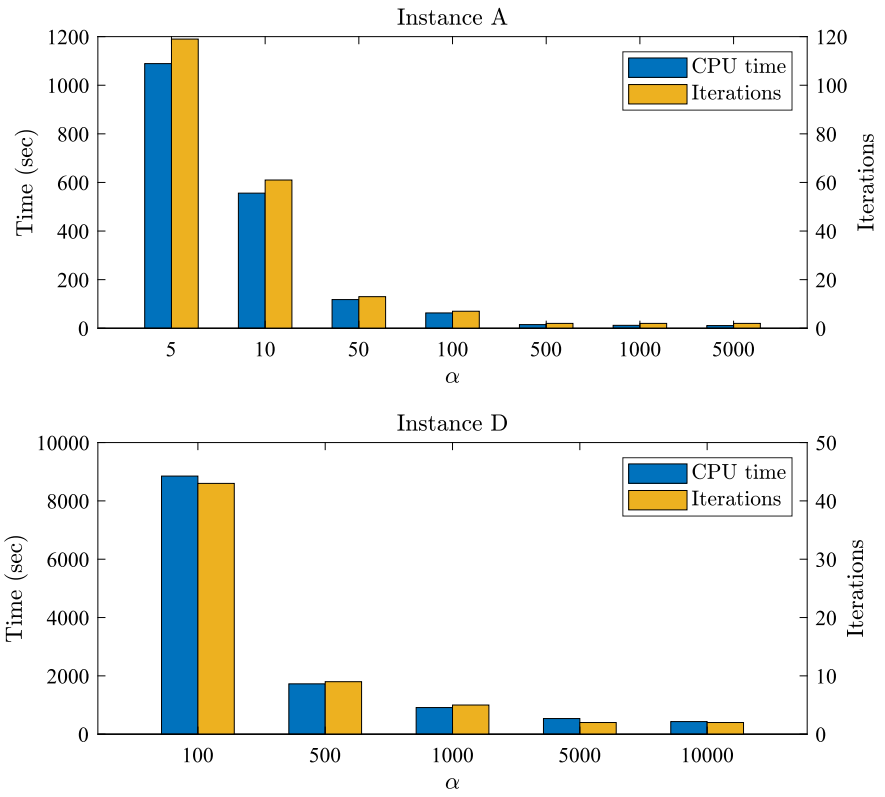


Fig. 4. Performance evaluation with α in SI decomposition.

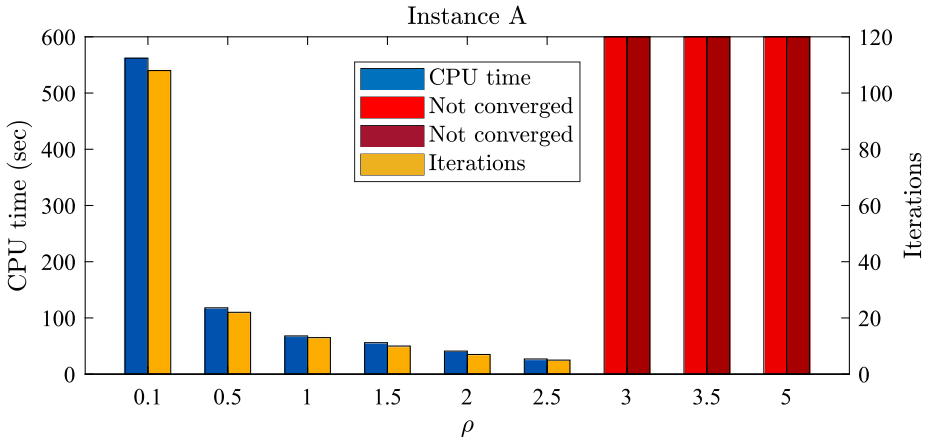


Fig. 5. Performance evaluation with ρ in PHA.

to reach optimal (even feasible) solutions at lower times, a tedious trial-and-error process is required to tune its step size ρ.

Although the proposed SI Decomposition algorithm could arrive at least at the same objective value as the monolithic approach in all tested instances, in other cases, large values of the weighing parameter α can result in an optimality gap. The presented analysis about the effect of the tuning parameters on the algorithm performance is preliminary. It is necessary to test the method on more two-stage scheduling problems of different nature in order to draw general conclusions.

Nonetheless, there are also some drawbacks. It is well known that arriving at a proven optimal solution is an extremely time consuming and computationally intensive task in large-scale mixed-integer problems. Therefore, high-quality solutions that can be obtained in reasonable amounts of time are usually preferred in the scheduling practice. The monolithic formulation naturally provides this feature by setting a time limit or an optimality gap as termination criteria in the MILP solver. However, the SI Decomposition cannot cope with either of such criteria: due to its nature, there is no guarantee of having a feasible solution until the algorithm has converged to $SI = 1$, and the number of needed iterations for this is not known beforehand. Nevertheless, and as opposed to other decomposition algorithms in the literature, convergence can be forced in a finite number of iterations if α is large enough and a feasible solution exists, of course.

The reader may note that decomposition algorithms can be sped up by not solving the subproblems to proven optimality (zero gap). However, this does not pose any bound on the actual optimality gap once a solution for the *global* problem is reached. Furthermore, relaxing the local optimality gap criterion too much causes oscillations in decomposition algorithms and hampers convergence.

The proposed decomposition algorithm always converged to a feasible and near-optimal solution in the tested case study, no matter the values set for the tuning parameters. Future work will focus on

extending the method to also handle continuous decision variables in a formal way, as well as on tackling continuous-time scheduling formulations.

CRediT authorship contribution statement

Daniel Montes: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **José Luis Pitarch:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **César de Prada:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Investigation, Funding acquisition, Conceptualization.

Data availability

No data was used for the research described in the article.

Acknowledgments

This research is funded by the Spanish MCIN/AEI through research projects a-CIDiT (PID2021-123654OB-C31, PID2021-123654OB-C32) and LOCPU (PID2020-116585GB-I00). The first author has received financial support from the 2020 call of the pre-doctoral contracts of the University of Valladolid, Spain, co-financed by Banco Santander.

References

Abouelrous, A., Gabor, A. F., & Zhang, Y. (2022). Optimizing the inventory and fulfillment of an omnichannel retailer: a stochastic approach with scenario clustering. *Computers & Industrial Engineering*, *173*, Article 108723. http://dx.doi.org/10.1016/J.CIE.2022.108723.

Baes, M., Oertel, T., & Weismantel, R. (2016). Duality for mixed-integer convex minimization. *Mathematical Programming*, *158*, 547–564.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, *4*(1), 238–252.

Birge, J. R. (1995). Models and model value in stochastic programming. *Annals of Operations Research*, *59*(1), 1–18.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, *3*(1), 1–122.

Carøe, C. C., & Schultz, R. (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters*, *24*(1–2), 37–45.

Colonetti, B., & Finardi, E. C. (2020). Combining Lagrangian relaxation, benders decomposition, and the level bundle method in the stochastic hydrothermal unit-commitment problem. *International Transactions on Electrical Energy Systems*, *30*(9).

CoPro (2020). Improved energy and resource efficiency by better coordination of production in the process industries. http://dx.doi.org/10.3030/723575, H2020-SPIRE-2016-723575. (Accessed 25 April 2024). https://www.aspire2050.eu/copro.

Flament, C. (1963). *Applications of Graph Theory to Group Structure*. Englewood Cliffs, NJ: Prentice-Hall.

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

Gower, J. C. (1985). Measures of similarity, dissimilarity and distance. *Vol. 5*, In *Encyclopedia of statistical sciences, johnson and CB read* (pp. 397–405). John Wiley and Sons.

Jiang, X., Bai, R., Wallace, S. W., Kendall, G., & Landa-Silva, D. (2021). Soft clustering-based scenario bundling for a progressive hedging heuristic in stochastic service network design. *Computers & Operations Research*, *128*, http://dx.doi.org/10.1016/J.COR.2020.105182.

Khalilabadi, S. M. G., Zegordi, S. H., & Nikbakhsh, E. (2020). A multi-stage stochastic programming approach for supply chain risk mitigation via product substitution. *Computers & Industrial Engineering*, *149*, Article 106786. http://dx.doi.org/10.1016/J.CIE.2020.106786.

Kim, K., & Zavala, V. M. (2018). Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs. *Mathematical Programming Computation*, *10*(2), 225–266.

Lee, M., Ma, N., Yu, G., & Dai, H. (2021). Accelerating generalized benders decomposition for wireless resource allocation. *IEEE Transactions on Wireless Communication*, *20*(2), 1233–1247. http://dx.doi.org/10.1109/TWC.2020.3031920.

Legrain, A., Omer, J., & Rosat, S. (2020). An online stochastic algorithm for a dynamic nurse scheduling problem. *European Journal of Operational Research*, *285*(1), 196–210.

Mitrai, I., & Daoutidis, P. (2022). A multicut generalized benders decomposition approach for the integration of process operations and dynamic optimization for continuous systems. *Computers & Chemical Engineering*, *164*, Article 107859.

Montes, D., Pitarch, J. L., & de Prada, C. (2022a). Decomposition of two-stage stochastic scheduling problems via similarity index. *Vol. 51*, In *Computer aided chemical engineering* (pp. 985–990). http://dx.doi.org/10.1016/B978-0-323-95879-0.50165-X.

Montes, D., Pitarch, J. L., & de Prada, C. (2022b). The similarity index to decompose two-stage stochastic scheduling problems. *IFAC-PapersOnLine*, *55*(7), 821–826. http://dx.doi.org/10.1016/j.ifacol.2022.07.546.

Palacín, C. G., Pitarch, J. L., Jasch, C., Méndez, C. A., & de Prada, C. (2018). Robust integrated production-maintenance scheduling for an evaporation network. *Computers & Chemical Engineering*, *110*, 140–151. http://dx.doi.org/10.1016/j.compchemeng.2017.12.005.

Palacín, C. G., Pitarch, J. L., Vilas, C., & de Prada, C. (2023). Integrating continuous and batch processes with shared resources in closed-loop scheduling: A case study on tuna cannery. *Industrial & Engineering Chemistry Research*, *62*(23), 9278–9289. http://dx.doi.org/10.1021/acs.iecr.3c00754.

Peng, Z., Zhang, Y., Feng, Y., Rong, G., & Su, H. (2019). A progressive hedging-based solution approach for integrated planning and scheduling problems under demand uncertainty. *Industrial & Engineering Chemistry Research*, *58*(32), 14880–14896.

Rockafellar, R., & Wets, R. J. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, *16*(1), 119–147.

Rudin, W., et al. (1976). *Vol. 3*, *Principles of mathematical analysis*. New York: McGraw-Hill.

Ruszczyński, A. (2003). Decomposition methods. In *Handbooks in operations research and management science*: *vol. 10*, *Stochastic programming* (pp. 141–211). Elsevier, http://dx.doi.org/10.1016/S0927-0507(03)10003-5.

Sand, G., & Engell, S. (2004). Modeling and solving real-time scheduling problems by stochastic integer programming. *Computers & Chemical Engineering*, *28*(6–7), 1087–1103.

Shor, N. Z. (1985). *Vol. 3*, *Minimization methods for non-differentiable functions* (1). (pp. 1–36).

Simkoff, J. M., & Baldea, M. (2020). Stochastic scheduling and control using data-driven nonlinear dynamic models: Application to demand response operation of a Chlor-Alkali plant. *Industrial & Engineering Chemistry Research*, *59*(21), 10031–10042.

Tanimoto, T. T. (1958). *Elementary mathematical theory of classification and prediction*: *Internal IBM technical report*, International Business Machines Corp..

Torres, J. J., Li, C., Apap, R. M., & Grossmann, I. E. (2022). A review on the performance of linear and mixed integer two-stage stochastic programming software. *Algorithms*, *15*(4), 103.

Watson, J.-P., & Woodruff, D. L. (2011). Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, *8*(4), 355–370.

Willett, P., Barnard, J. M., & Downs, G. M. (1998). Chemical similarity searching. *Journal of Chemical Information and Computer Sciences*, *38*(6), 983–996. http://dx.doi.org/10.1021/ci9800211.