A parallel Branch-and-Fix Coordination based matheuristic algorithm for solving large sized multistage stochastic mixed 0-1 problems

Unai Aldasoro[a], Laureano F. Escudero[b], María Merino[c], Gloria Pérez[c]

[a]*BCAM - Basque Center for Applied Mathematics, Spain. ualdasoro@bcamath.org*

[b]*Estadística e Investigación Operativa, Universidad Rey Juan Carlos, URJC, Spain. laureano.escudero@urjc.es*

[c]*Matemática Aplicada, Estadística e Investigación Operativa, Universidad del País Vasco, UPV/EHU, Spain. maria.merino@ehu.es, gloria.perez@ehu.es*

Abstract

A parallel matheuristic algorithm is presented as a spin-off from the exact Branch-and-Fix Coordination (BFC) algorithm for solving multistage stochastic mixed 0-1 problems. Some steps to guarantee the solution's optimality are relaxed in the BFC algorithm, such that an incomplete backward branching scheme is considered for solving large sized problems. Additionally, a new branching criterion is considered, based on dynamically-guided and stage-wise ordering schemes, such that fewer Twin Node Families are expected to be visited during the execution of the so-called H-DBFC algorithm. The inner parallelization IH-DBFC of the new approach, allows to solve in parallel scenario clusters MIP submodels at different steps of the algorithm. The outer parallel version, OH-DBFC, considers independent paths and allows iterative incumbent solution values exchanges to obtain tighter bounds of the solution value of the original problem. A broad computational experience is reported for assessing the quality of the matheuristic solution for large sized instances. The instances dimensions that are considered are up to two orders of magnitude larger than in some other works that we are aware of. The optimality gap of the H-DBFC solution value versus the one obtained by a state-of-the-art MIP solver is very small, if any. The new approach frequently outperforms it in terms of solution's quality and computing time. A comparison with our Stochastic Dynamic Programming algorithm is also reported. The use of parallel computing provides, on one hand, a perspective for solving very large sized instances and, on the other hand, an expected large reduction in elapsed time.

Keywords: Multistage stochastic mixed 0-1 optimization, matheuristic, Branch-and-Fix Coordination, break stage scenario clustering, parallel computing, message-passing interface

1. Introduction

A multistage stochastic mixed-integer optimization model has a more complex scenario information structure than its related, sometimes approximate, two-stage model. Moreover, there have not been too many attempts to solve large sized general multistage stochastic mixed 0-1 models up to optimality, due to their complexity, see [41, 49, 50], among others. Thus, stochastic programs for many real-life instances require intensive computational effort. The solution has to be obtained by using decomposition algorithms that exploit

[1]Corresponding author. María Merino.
Tel.: +34 946012523; fax: +34 946012516
E-mail address: maria.merino@ehu.es
Address: Barrio Sarriena s/n, 48940 Leioa, Bizkaia, Spain

the nice structure of models based on scenario analysis and convexity. Those algorithms and others can be classified in the following types for two-stage and multistage problems:

1. Benders Decomposition (BD) methodology [8]. Probably, it is the most used exact methodology for solving two-stage problems with continuous variables in the second stage. The L-Shaped algorithm [69] is the well-known first published algorithm on the subject. See also [5, 51], among many others. The (nested) version for multistage problems, by construction, does not guarantee the optimality of the solution for problems with integer variables in any stage, but the first one. However, it is one of the best known algorithms for linear problems, see the seminal work [10], recently [75], among others.

2. Two-stage Lagrangean Decomposition (LD) heuristic methodology. It is another very interesting methodology for problem solving in mathematical optimization. The main aim of LD consists of providing (hopefully, strong) bounds (in case of minimization) on the solution value of a problem, such that the goodness of a feasible solution obtained by other means can be assessed. However there are well-known conditions (see [34, 35]) under which the own Lagrangean solution can be declared an optimal one. Furthermore, if it is not an optimal one, it can be used by an ad-hoc algorithm for obtaining a feasible solution, based on the scheme that fixes appropriate variables to some of their Lagrangean values. Afterwards, the resulting model is solved, such that it has usually either an structure to take benefit from it or can be decomposed in submodels that are also easy enough to solve.

3. Multistage Clustering Lagrangean Decomposition (MCLD) heuristic methodology. There are very few LD approaches for solving multistage stochastic mixed 0-1 problems. See our approach [26] for obtaining strong lower bounds to the solution values of small to medium sized instances, where the scenarios are distributed in clusters (also so-called bundles). The distribution reduces in a systematic way the number of NAC (non-anticipativity constraints that equate the scenario cluster variables in the splitting formulation of the model). The scheme is based on a so-called break stage, such that the NAC of the variables to be dualized belong to the nodes in the scenario tree whose related stages are up to the break one. See also [24] as a specialization of [26] for providing a Lagrangean heuristic solution for the multistage stochastic pure 0-1 version of the difficult deterministic (combinatorial) facility location - assignment problem. The quasi-optimality gap that has been obtained for large sized instances is very small. This type of MCLD approach has also been considered in [25] for obtaining strong lower bounds on the solution value of a problem with a mixture of the first- and second-order time stochastic dominance (TSD) risk averse measure. Other types of multistage LD algorithms, see [52], also split the scenario tree in blocks, so that the root node is also connected to the leaf ones by considering NAC of some variables. However, the related nodes do not necessarily belong to the same stage. NAC dualization allows to obtain lower bounds to the problem and, from there, heuristic solutions are obtained by an ad-hoc scheme. The NAC (to dualize) are chosen based on the scenario tree structure of the problem to solve.

4. Regularization. To the best of our knowledge, an interesting approach for speeding up the convergence of LD approaches for solving mathematical optimization linear models was first introduced in [53, 62], and coined as Augmented Lagrangean Decomposition (ALD). The iterative approach considers the splitting variable representation of the original model, such that the constraints equating the replicas of the variables are dualized, and the square terms of those constraints are appended to the Lagrangean function; its convergence is proven. Some computational experience is reported in [53] by solving quadratic scenario submodels using a nonlinear interior point algorithm. Recently, [45] and others have extended the ALD approach to stochastic optimization problems, where the constraints equating the replicas of some variables in some scenarios are precisely the NAC of each splitting variable. An extension of the (convex) quadratic terms for reducing an excessive oscillation of the successive solutions (and, then, speeding up the convergence of the algorithm) has been coined as the regularization term. The methodology has been known regularized decomposition. The quadratic regularization for

stochastic linear optimization was first developed in [63] for two-stage problems, see also [66] for two-stage and multistage problems, and [6, 7] for multistage problems, among others. The Stochastic Dual Dynamic Programming (SDDP) approach [56, 57] is a decomposition methodology that has been most frequently used for testing different regularization mechanisms, since there it is assumed the stage-wise independence of the random process and, then, the dimensions of the scenario tree could still be manageable. The mechanism that is considered in [6, 63] for the stage-wise independence of the random process consists of appending to the objective function the (convex) quadratic regularization function. This function is based on the difference between the variables in the nodes of the tree and their values in the incumbent solution of the model. However, for a high number of periods in the time horizon or, as it frequently happens in practice, the outcome of the random parameters at a given stage is not independent of the outcomes of the previous stages, then, the number of regularized quadratic terms could be unmanageable. For that type of problems, see in [6] its Algorithm 2 where the incumbent solutions are not indexed by the nodes of the scenario tree. So, the reference for the quadratic function could have different parts of the left hand side of constraints related to the stages and not to the nodes in the tree. Faster convergence is proved.

5. The Progressive Hedging algorithm (PHA) for multistage primal decomposition was introduced in [60] as a specific regularization approach for solving up to optimality multistage stochastic linear problems. It is broadly used and, recently it has been extended [72] to also considering integer variables. However, by construction, the optimality of the incumbent solution cannot be guaranteed any more. The basic idea of PHA is as follows: (a) Decompose the original model by scenarios (i.e., relaxing the NAC of variables in all nodes in the scenario tree) and solve each submodel, where a (linear) penalization term and its (quadratic) regularization have been added to each scenario function; (b) Obtain an implementable solution (i.e., a solution satisfying the NAC) that probably is not a feasible one (i.e., an admissible solution) by averaging for each node in the tree the related scenario solutions. Notice that now, by construction, the NAC are satisfied, but likely some of the other constraints are not; (c) Update the (linear) penalization term for each scenario submodel by using a subgradient estimator type of the non-implementable solution (i.e., the difference of the scenario solution with respect to the current implementable one) for each node; (d) Append the penalization term to each scenario submodel plus its weighted square function; (e) Iterate until a stopping criterion is satisfied. We conjecture that the convergence of PHA could be speed up by building scenario cluster submodels, instead of single scenario ones, by using the break stage based scheme presented in [26]. See in [33] an extension of the algorithm for two-stage and multistage by using by using a scenario cluster (bundle is called there) approach. It allows to obtain lower bounds of the optimal mixed integer solution and, then, the goodness of the solution can be assessed. Moreover, in that case, the approach would not be too far from one of the Lagrange multipliers updating schemes used in that other algorithm, aside that the objective function should have (convex) quadratic terms for mixed integer large sized problems. Limited computational experience for two-stage is reported. The multistage case is still a challenge.

6. Multistage Stochastic Dynamic Programming (SDP). It is a variant of the SDDP methodology mentioned above, for solving large sized Risk Neutral (RN) multistage stochastic linear problems with stage-wise independent uncertainty. See [56, 57, 61]. Some works in the literature allow to consider Markovian multistage stochastic problems, where the uncertainty in each node of a given stage depends on the history of its ancestor nodes, see [6, 30, 75]. The treatment of the Conditional Value-at-Risk (CVaR) risk averse measure in SDP was introduced in [36, 42, 68]. Recently, some works consider mixed-integer problems, see [3, 16] and, even time-consistent and time-inconsistent stochastic dominance risk averse measures, see [31, 32]. Given the integer character of some variables, the usage of SDP for problem solving with integer variables in any stage, cannot guarantee the solution's optimality but, in any case, they can deal with very large sized instances.

7. Multistage scenario cluster primal decomposition. This type of algorithms is very promising, since the

related decomposition is performed on clusters of scenarios, seeking for a (hopefully, good) feasible solution. Some of those algorithms guarantee the optimality of the incumbent solution. Specifically, see [14, 22, 23, 29, 64, 74, 75] for decomposition approaches that consider scenario clustering for solving large sized multistage stochastic mixed integer problems. The so-called Branch-and-Fix Coordination (BFC) methodology is used in the decomposition approaches presented in [22, 23, 29, 55] to generate independent scenario clusters, such that multiplicity of any scenario is not allowed among the clusters. The BFC methodology is an exact one that relaxes a given subset of non-anticipativity constraints (NAC) from the model (up to the break stage mentioned above), but it takes care of them in the execution of the algorithm. The bounding method presented in [64], as an extension and improvement of the two-stage bounding method introduced in [65], splits the scenario set in clusters. For that purpose, the NAC are relaxed in the nodes of the tree, so that the related scenarios are grouped in different clusters. Notice that this primal based algorithm splits the scenario tree in a manner that is close (conceptually, at least) to the scheme used in the dual algorithm presented in [52]. The scenario multiplicity in the clusters allows to obtain strong lower bounds. On the other hand, a scheme for fixing the solution of some scenario clusters allows to obtain feasible solutions and, then, upper bounds on the solution value of the original model can be computed. The bounding method presented in [74] decomposes the scenario tree into a number of smaller trees. Vertex cuts are used for that purpose and, then, the scenarios are clustered depending on the stages used for the cuts, so that the root in the tree is separated from the leaves. Lower bounds are computed as the weighted sum of the solution values of the subproblems associated with the subtrees that are built (i.e., a NAC relaxation is performed on the nodes of the scenario tree related to the vertex cuts). Upper bounds are obtained by fixing the root-to-cut variables at their best values.

Parallel Computing (PC) offers an alternative for solving very large sized problems by parallelizing the execution of the MIP submodels that appear in the decomposition algorithms. Currently, at hardware level, PC is mainly based on clusters and multicore processors. For basic information, see [17, 37], among others. The nature of the cooperation between processors can differ depending on the way in which processors exchange information. One of the parallel architectures is distributed memory, managed by *message passing* (for example, using Message Passing Interface, MPI). Over the last two decades papers on stochastic optimization have appeared in the relevant literature that use PC for two-stage and multistage stochastic linear as well as mixed 0-1 optimization, see e.g., [1, 9, 46, 21] and references therein. Recently, parallel computing versions of the exact BFC methodology was presented in [4, 55]. See also the parallel matheuristic bounding methods presented in [3, 6, 53, 61, 64, 74] that also allow large sized instances to be solved.

This work presents several strategies in order to improve the performance of decomposition algorithms for solving large sized multistage stochastic mixed 0-1 problems. In particular, we introduce an algorithm so-called *Dynamically-guided and stage-ordered Branch-and-Fix Coordination* algorithm (for short, DBFC) that, belonging to the multistage BFC methodology, strongly improves the performance of previous BFC algorithms presented in [4, 22] and references therein. The main improvements are in the dynamic branching mechanism that allows to solving much larger sized multistage stochastic mixed 0-1 problems, they are as follows: (1) The scenario cluster partitioning is based on the so-called break stage. It is one of the key elements in the new approach and considers stage-wise variable ordering in the problem and auxiliary scenario cluster submodels; (2) The selection of the stage-wise ordered branching variable in the Branch-and-Fix (BF) nodes (i.e., scenario cluster based Brand-and-Bound nodes to be jointly handled) jumps over the variables that are currently satisfying their non-anticipativity constraints (NAC); (3) Dynamically-guided branching in the selection of the 0-1 direction to be considered first. It is based on the frequency of the values of the branching variable in the solution of the previous submodels of the scenario clusters to which the variable belongs to; (4) The solution of the scenario cluster submodels is frequently required for the same branching path, so, the submodels are only solved once and stored for later use; and (5) Different types of submodels are considered for obtaining feasible solutions by fixing a variety of variables at the values obtained in previous steps.

Additionally, as very large sized instances are not expected to be solved efficiently up to optimality, a matheuristic algorithm, so-called H-DBFC is proposed, based on the new approach, DBFC. Its different strategies are intended to guarantee an harmony between solution's quality and computational effort (in memory and computing time) as it can be assessed in the computational experience whose results are reported in the paper. It has the advantage of being applicable to any stochastic mixed 0-1 optimization problem without the high model dependence exhibited by some Stochastic Dynamic Programming (SDP) schemes, see [16]. However, it is not yet able to solve the gigantic sized instances solved by the SDP algorithm, since its lower decomposition capabilities lead to memory limit issues.

Another important contribution of this work is the extension of the parallel versions of H-DBFC presented in [2], so-called inner and outer ones, to the matheuristic environment, where a triple thread assignment is considered for the thread hierarchy. The inner parallel version provides a perspective for solving very large sized instances. It allows to solve in parallel scenario cluster MIP submodels at different steps of the algorithm. Due to the features of the new H-DBFC approach, fewer cluster submodels are solved by each iteration, on average; therefore, a smaller number of threads is needed to achieve an equivalent work balance as in the previous paper. The outer parallelization presented here has been adapted to deal with a dynamic root node matching criterion, as opposed to the deterministic one of the old algorithm. It allows for seeking better solutions since it uses an iterative dynamical scheme for generating paths of partial variable branchings. Then, tighter bounds of the original problem's solution value can also be obtained. Both versions allow to solve problems up to several millions constraints, over half a million 0-1 variables and a couple of million continuous variables with a very small quasi-optimality tolerance in an affordable elapsed time.

The rest of the paper is organized as follows: Section 2 presents the basic models and scenario cluster submodels used in the multistage DBFC algorithm. Section 3 introduces the algorithm as well as the aim and perspective of the proposed spin-off matheuristic H-DBFC. Section 4 presents the inner and outer parallel versions of H-DBFC. Section 5 reports the main results of a broad computational experience to assess the validity of H-DBFC to solve large sized problems for the serial and parallel versions of the algorithm. The computational results are compared with plain use of CPLEX and the serial and parallel versions of our SDP algorithm. Finally, Section 6 concludes and outlines future work.

2. Multistage stochastic mixed 0-1 models

For the general formulation of a multistage model, where decisions have to be made in a stage-wise manner, let Ω denote the finite set of scenarios that are considered to be representative of the uncertainty quantification in the problem and $\mathscr{T}$ is the set of stages in a given time horizon (where $T \equiv |\mathscr{T}|$ is the last stage). Let a multistage scenario tree to represent the uncertainty, where $\mathscr{G}$ is the set of nodes in the scenario tree and $\mathscr{G}^t \subseteq \mathscr{G}$ is the set of nodes that belong to stage t, for $t \in \mathscr{T}$. Let also g, for $g \in \mathscr{G}^t$, denote a node in the tree, such that $t(g) \in \mathscr{T}$ gives the stage to which node g belongs to; and $\Omega^g \in \Omega$ is the set of scenarios that belong to group g (with a one-to-one correspondence with node g in the scenario tree) that have an identical realization of their uncertain parameters up to stage $t(g)$. Let $\tilde{\mathscr{A}}^g$ and $\mathscr{S}^g$ denote the set of ancestor nodes in the tree to node g (including itself), and the set of successor nodes to node g, respectively.

It is worth to point out that it is also known in node g what scenarios will not happen in the future, i.e., the scenarios in set Ω/Ω^g. So, the decision variables in each node g should, thus, be based on the known information (given by set $\tilde{\mathscr{A}}^g$) on one hand and without anticipating future events on the other hand, although using as much information as possible, given by set $\mathscr{S}^g$, for $g \in \mathscr{G} : t(g) < T$. That is, the extension of the non-anticipativity principle introduced in [73] for two-stage problems should be satisfied.

Without loss of generality, consider the compact representation of the multistage stochastic mixed 0-1

model for minimizing the objective function expected value over the set of scenarios Ω,

$$\begin{aligned} z^{DEM} = \quad & \min \sum_{g \in \mathscr{G}} w^g (a^g x^g + b^g y^g) & \\ \text{s.t.} \quad & \sum_{q \in \mathscr{A}^g} (A_g^q x^q + B_g^q y^q) = h^g & \forall g \in \mathscr{G} \\ & x^g \in \{0,1\}^{nx(g)}, y^g \in \mathbb{R}^{ny(g)} & \forall g \in \mathscr{G}, \end{aligned} \tag{1}$$

where w^g is the weight of node (i.e., probability of the scenario group) g to be computed as $\sum_{\omega \in \Omega^g} w^\omega$; ω is a scenario in set Ω^g; w^ω is the modeler-driven weight assigned to scenario $\omega \in \Omega$, where $\sum_{\omega \in \Omega} w^\omega = 1$; $\mathscr{A}^g \subseteq \tilde{\mathscr{A}}^g$ is the set of indexes for the ancestor nodes of node g (including itself) whose decision variables have direct influence (i.e., have non zero elements) on the constraints in node g, where $\mathscr{A}^1 = \{1\}$; x^g and y^g are the vectors of the 0-1 and continuous variables for node g, respectively; a^g and b^g are the vectors of the objective function coefficients for the 0-1 and continuous variables, respectively; A_g^q and B_g^q are the constraint matrices of the ancestor node $q \in \mathscr{A}^g$ in node g for the vectors x^q and y^q, respectively; h^g is the right-hand-side vector (rhs) for node g; and $nx(g)$ and $ny(g)$ are the number of 0-1 and continuous variables, respectively, for $g \in \mathscr{G}$, $nx = \sum_{g \in \mathscr{G}} nx(g)$ and $ny = \sum_{g \in \mathscr{G}} ny(g)$, such that it is assumed that $nx(g)+1$ and $ny(g)+1$ are the numberings of the first 0-1 and continuous variables for node $g+1 \in \mathscr{G}$, respectively. Observe that Ω^g is a singleton set for $g \in \mathscr{G}^T$. See [11, 39, 40, 58, 59, 67, 71], among others, for the main concepts on stochastic optimization via scenario tree analysis. Notice that model (1) is so-called Risk Neutral (RN) model.

2.1. Scenario clustering

In [22] we propose a decomposition of the scenario tree into a set of subtrees. Based on this cluster decomposition concept a mixture of the splitting and compact representations of the original multistage stochastic mixed 0-1 RN model (1) is presented. The reason for this decomposition is based on the way in which our BFC decomposition algorithm works. It explicitly considers the NAC of the variables of the nodes in different cluster subtrees. By construction, those nodes belong to stages up to a given so-called break stage, t^* (see below). On the other hand, the NAC of the variables in the nodes that belong to the stages from t^*+1 until the last one are implicitly considered while solving the scenario cluster submodels. For completeness, let us consider the following definitions taken from [22].

Definition 1. *A **break stage**, say t^*, is a modeler-driven stage from set $\mathscr{T}$ such that the number of scenario clusters is $|\mathscr{C}| = |\mathscr{G}^{t^*+1}|$. In this case, any scenario cluster indexed with c, for $c \in \mathscr{C}$, is induced by a scenario tree node, say g_c from set $\mathscr{G}^{t^*+1}$, and it contains all the scenarios belonging to group Ω^{g_c} (that has a one-to-one correspondence with node g_c in the scenario tree).*

Definition 2. *The **scenario cluster** submodels are those that result from the relaxation of the NAC in model* (1) *in the nodes that belong to stages up to break stage t^*.*

Let us first split the set of stages $\mathscr{T}$ in two subsets, such that $\mathscr{T} = \mathscr{T}_1 \bigcup \mathscr{T}_2$, where $\mathscr{T}_1 = \{1, \ldots, t^*\}$ and $\mathscr{T}_2 = \{t^*+1, \ldots, T\}$. Once the break stage t^* is decided, the corresponding t^*-cluster partition is given and, then, the number of scenario clusters in $\mathscr{C}$ is fixed to $|\mathscr{G}^{t^*+1}|$, i.e., each node in $\mathscr{G}^{t^*+1}$ belongs to just one cluster in set $\mathscr{C} = \{1, ..., C\}$, with $C = |\mathscr{C}|$. For a node g_c in set $\mathscr{G}^{t^*+1}$, let $\Omega_c = \Omega^{g_c}$ denote the set of scenarios in the tree and $\mathscr{G}_c \subseteq \mathscr{G}$ is the set of nodes in cluster $c \in \mathscr{C}$, where a node g belongs to set $\mathscr{G}_c$ provided that $\Omega^g \cap \Omega_c \neq \emptyset$; x_c^g and y_c^g denote the replicas of the variables in vectors x^g and y^g for node $g \in \mathscr{G}_c$ in cluster $c \in \mathscr{C}$, respectively; and x_c and y_c are the vectors that include the set of variables in the vectors x_c^g and y_c^g for all nodes $g \in \mathscr{G}_c$ in cluster $c \in \mathscr{C}$, respectively. Notice that there is only one replica for $t(g) \in \mathscr{T}_2$ and there are $|\mathscr{C}|$ replicas for $t(g) \in \mathscr{T}_1$, at most. Properties: $\mathscr{G}^t \cap \mathscr{G}_c$ is singleton for $t \in \mathscr{T} : t \leq t^*+1$; let $\mathscr{C}^g$ denote the set of scenario clusters where a scenario, at least, belongs to group Ω^g, for $g \in \mathscr{G}$, so, $\mathscr{C}^g$ is a singleton for $t(g) \in \mathscr{T}_2$; $c \in \mathscr{C}^g \equiv g \in \mathscr{G}_c$ for $g \in \mathscr{G}$; $\Omega_c \subseteq \Omega^g$ for $g \in \mathscr{G}_c \bigcap \mathscr{G}^t$, $t \in \mathscr{T}_1$, $c \in \mathscr{C}$; and $\Omega^g \subseteq \Omega_c$ for $g \in \mathscr{G}_c \bigcap \mathscr{G}^t$, $t \in \mathscr{T}_2$, $c \in \mathscr{C}$.

Now, the scenario cluster submodel can be expressed in compact representation, for $c \in \mathscr{C}$,

$$\begin{aligned} z_c = \quad & \min \sum_{g \in \mathscr{G}_c} w_c^g (a^g x_c^g + b^g y_c^g) \\ \text{s.t.} \quad & \sum_{q \in \mathscr{A}^g} A_g^q x_c^q + B_g^q y_c^q = h^g \qquad \forall g \in \mathscr{G}_c \\ & x_c^g \in \{0,1\}^{nx(g)}, y_c^g \in \mathbb{R}^{ny(g)} \quad \forall g \in \mathscr{G}_c, \end{aligned} \tag{2}$$

where $w_c^g = \sum_{\omega \in \Omega^g \cap \Omega_c} w^\omega$ for $g \in \mathscr{G}_c$ and, so, $w_c^g = w^{g_c}$ for $g : t(g) \in \mathscr{T}_1$ and $w_c^g = w^g$ for $g : t(g) \in \mathscr{T}_2$.

Observe that in the splitting-compact representation of the original model (1), the nonanticipativity principle is implicitly taken into account for the stages $t \in \mathscr{T}_2$, since the submodel for each cluster is formulated via a compact representation. On the other hand, the (explicit) NAC of the variables in the scenario tree nodes that belong to the stages in set $\mathscr{T}_1$ can be formulated by observing that the clusters c and c' have node g in common (i.e., $g \in \mathscr{G}_c \cap \mathscr{G}_{c'}$) if and only if $g \in \mathscr{G}^t : t \in \mathscr{T}_1$. So, the cluster submodels (2) are linked by the NAC to be expressed,

$$x_c^g - x_{c'}^g = 0 \quad \forall c, c' \in \mathscr{C} : c \quad c',\ g \in \mathscr{G}_c \cap \mathscr{G}_{c'} \tag{3}$$

$$y_c^g - y_{c'}^g = 0 \quad \forall c, c' \in \mathscr{C} : c \quad c',\ g \in \mathscr{G}_c \cap \mathscr{G}_{c'}. \tag{4}$$

Model (1) can be represented by a mixture of the splitting variable representation (for explicitly satisfying the NAC between the cluster submodels) and the compact representation (for implicitly satisfying the NAC of each cluster submodel). So, the cluster splitting-compact representation can be expressed

$$\begin{aligned} z^{DEM} = \quad & \min \sum_{c \in \mathscr{C}} \sum_{g \in \mathscr{G}_c} w_c^g (a^g x_c^g + b^g y_c^g) \\ \text{s.t.} \quad & \sum_{q \in \mathscr{A}^g} A_g^q x_c^q + B_g^q y_c^q = h^g \qquad \forall g \in \mathscr{G}_c, c \in \mathscr{C} \\ & \text{NAC } (3) - (4) \\ & x_c^g \in \{0,1\}^{nx(g)}, y_c^g \in \mathbb{R}^{ny(g)} \quad \forall g \in \mathscr{G}_c, c \in \mathscr{C}. \end{aligned} \tag{5}$$

It is worth to point out that the efficiency of a MIP engine for solving model (5) is very low, but it paves the way for performing model decomposition, see below.

Note: As a technical point to be used through this work, let us consider that the numbering of the nodes in the scenario tree is consecutive, and if g' is the first lexicographically ordered node in set $\mathscr{G}^t$, then $g'-1$ is the last lexicographically ordered node in set $\mathscr{G}^{t-1}$. Moreover, $\mathscr{I}_x^g$ and $\mathscr{I}_y^g$ denote the set of indexes of the variables in vector x^g and y^g, respectively, for $g \in \mathscr{G}$, and $(x^g)_i$ and $(y^g)_i$ are the i-th variables in the vectors corresponding to the set of indices $\{nx(g-1)+1,\ldots,nx(g)\} \subset \{1,\ldots,nx\}$ and $\{ny(g-1)+1,\ldots,ny(g)\} \subset \{1,\ldots,ny\}$, respectively.

2.2. Main concepts of the Branch-and-Fix Coordination methodology

For completeness, let us review some concepts of the exact Branch-and-Fix Coordination(BFC) methodology, see [22], where DBFC, a substantial improvement, has been derived from.

Definition 3. *A* Branch-and-Fix *(BF) tree, say BF_c, associated with scenario cluster indexed with c is a Branch-and-Bound tree (B&B) for solving the MIP submodel* (2)*, but in a coordinated way with the submodels of the other clusters in set $\mathscr{C}$, since the NAC* (3)-(4) *must be satisfied.*

Definition 4. *Two 0-1 variables, say, $(x_c^g)_i$ and $(x_{c'}^g)_i$, for $i \in \mathscr{I}_x^g$, are said to be* common variables *for the scenario clusters c and c', for $c, c' \in \mathscr{C} : c \quad c'$, $i \in \mathscr{I}_x^g$, $g \in \mathscr{G}^t$, $t \in \mathscr{T}_1$, provided that $g \in \mathscr{G}_c \cap \mathscr{G}_{c'}$.*

As an additional notation, let $\mathscr{N}_c$ be the set of active nodes in BF_c, $c \in \mathscr{C}$.

Definition 5. *Any two BF nodes, say, $a \in \mathcal{N}_c$ and $a' \in \mathcal{N}_{c'}$ for $c \quad c'$ are said to be* twin nodes *with respect to a given scenario tree node if their paths from the root nodes to each of them in their trees BF_c and $BF_{c'}$ are such that some of their common variables $(x_c^g)_i$ and $(x_{c'}^g)_i$, have been branched on or fixed at the same 0-1 value, for $c, c' \in \mathscr{C} : c \quad c', i \in \mathscr{I}_x^g, g \in \mathscr{G}_c \cap \mathscr{G}_{c'}$.*

Definition 6. *A* Twin Node Family *(for short, TNF) $\mathscr{J}_f$ is a set of nodes such that any node is a* twin *node to all the other node members of the family, for $f \in \mathscr{F}$, where $\mathscr{F}$ is the set of the families.*

Definition 7. *A TNF is said to be a* candidate TNF *(for short, cTNF) if there is one common variable, at least, in each of their BF node members that has not been yet branched on, nor fixed at 0-1 values.*

Definition 8. *A TNF is said to be an* integer TNF *if all common variables in the TNF have already been branched on or fixed at (obviously, the same) 0-1 values and their related NAC* (3) *are satisfied (see below).*

The branching in each tree BF_c for $c \in \mathscr{C}$ in the BFC methodology is only performed on the common variables, such that once done, a feasible solution for the scenario clusters can be obtained by solving the submodels (2) $\forall c \in \mathscr{C}$ where the common variables are fixed at the branched on 0-1 values. Notice that x,y-solution is a feasible one for the original model (5) provided that the NAC (3)-(4) are satisfied. The feasibility of NAC (3) can be obtained by coordinating the branching on the common variables such that the i-th variable is equated $(x_c^g)_i = x_{c'}^g = 0, \forall c, c' \in \mathscr{C} : c \quad c', \ g \in \mathscr{G}_c \cap \mathscr{G}_{c'}$ for all $i \in \mathscr{I}_x^g, g \in \mathscr{G}^t, t \in \mathscr{T}_1$ on the 0-1 branched values. See in [22] the details of the BFC algorithm.

Remark 1. *A BF tree differs from a single B&B tree associated with a scenario cluster in that the fixing of a 0-1 common variable in a B&B node (in our case, BF node), say node c, automatically produces the fixing of the replica of that variable to the same value in all nodes in the TNF under consideration, say, $\mathscr{J}_f$ such that $c \in \mathscr{J}_f$, and the implications could go further.*

Remark 2. *The BFC methodology does not consider BF nodes in an individual basis and, then, it does only consider TNFs, so, through the rest of the paper the term node is only used for scenario tree nodes.*

2.3. Auxiliary models to be used in DBFC

Let us introduce some additional notation:

- $\mathscr{I}_x$, set of indices of the common x-variables. They are to be branched on or fixed at in the BF trees, such that $\mathscr{I}_x = \cup_{g \in \mathscr{Q}} \mathscr{I}_x^g$, where $\mathscr{Q} \equiv \{g \in \mathscr{G} : t(g) \in \mathscr{T}_1\}$. Analogously, $\mathscr{I}_y = \cup_{g \in \mathscr{Q}} \mathscr{I}_y^g$.
- $\bar{i}$, index of the branching (common) variable, for $\bar{i} \in \mathscr{I}_x$, to be performed at Steps 2 and 3 of H-DBFC, see Algorithm 1 introduced in Section 3.3, so that a given cTNF is considered at Step 4.
- $\mathscr{I}_1 \subseteq \mathscr{I}_x$, set of indices of the already branched on or fixed at common variables at a given cTNF. Let the set of common variables be denoted by $\{x_i : i \in \mathscr{I}_1\}$. Notice that it is assumed that those variables are step-wise lexicographically ordered.
- $\hat{x}_c^g$, the 0-1 value vector of vector x_c^g, if any, for scenario cluster $c \in \mathscr{C} : g \in \mathscr{G}_c$, such that $t(g) \in \mathscr{T}_1$, thus $\hat{x}^g = \hat{x}_c^g$ provided that the NAC (3) are satisfied.

The pruning scheme at a cTNF is done by comparing the incumbent value, say, $\overline{z}^{DEM}$ for model (1) and $z = \sum_{c \in \mathscr{C}} z_c$, where z_c is the solution value of submodel (2) for scenario cluster c that has been obtained in the branching iteration of the last branched on variable x_i, for $x_i \in \mathscr{I}_1$, where $i \in \mathscr{I}_x^g$ for $g \in \mathscr{G}_c$.

Notice that for the cTNF defined by branching on variable $x_{\bar{i}}$, the NAC (3) in model (5) are already algorithmically satisfied for the branched on or fixed variables $x_i, \forall i \in \mathscr{I}_1$. After the cluster submodels (2) are solved at Step 4 of H-DBFC (see Section 3.3), the NAC (3) of the variables $(x^g)_i, \forall i \in \mathscr{I}_x^g, g \in \mathscr{G}^t, t \in \mathscr{T}_2$ are

implicitly satisfied. Additionally, if the testing of the NAC of the variables $x_i, \forall i \in \mathscr{I}_x \setminus \mathscr{I}_1$ is positive (i.e., they are satisfied), then, an integer TNF has been obtained.

At any integer TNF a feasible solution for the original model (1) could be obtained (in Step 5 of H-DBFC) by fixing all the $x-$variables at their current 0-1 values, such that the LP model can be expressed

$$\begin{array}{rll} z_{LP}^{TNF} = & \min \sum_{g\in\mathscr{G}} w^g (a^g x^g + b^g y^g) & \\ \text{s.t.} & \sum_{q\in\mathscr{A}^g} (A_g^q x^q + B_g^q y^q) = h^g & \forall g \in \mathscr{G} \\ & x^g = \hat{x}^g & \forall g \in \mathscr{G} \\ & y^g \in \mathbb{R}^{ny(g)} & \forall g \in \mathscr{G}. \end{array} \tag{6}$$

It is worth to point out that the LP model (6) can have very high dimensions. For those instances, given the nice structure of the model, it can be t^*-based Lagrangean decomposed as in [26], where a quadratic regularization term can be appended to speed up the convergence of the approach. See in Section 6 our future research plans on the subject. Notice that if the model is feasible then the new incumbent solution value is $\overline{z}^{DEM} := \min\{z_{LP}^{TNF}, \overline{z}^{DEM}\}$.

Let $(\hat{x}^g, \hat{y}^{g,TNF})$ denote the solution of model (6) $\forall g \in \mathscr{G}^t, t \in \mathscr{T}_1$. Observe that the related NAC (3) and (4) are satisfied. Additionally, a new auxiliary MIP submodel (7) is defined by fixing the vectors x^g and y^g at the values in $\hat{x}^g$ and $\hat{y}^{g,TNF}$, respectively.

$$\begin{array}{rll} z_f^{TNF} = & \min \sum_{g\in\mathscr{G}} w^g (a^g x^g + b^g y^g) & \\ \text{s.t.} & \sum_{q\in\mathscr{A}^g} (A_g^q x^q + B_g^q y^q) = h^g & \forall g \in \mathscr{G} \\ & x^g = \hat{x}^g, y^g = \hat{y}^{g,TNF} & \forall g \in \mathscr{G}^t, t \in \mathscr{T}_1 \\ & x^g \in \{0,1\}^{nx(g)}, y^g \in \mathbb{R}^{ny(g)} & \forall g \in \mathscr{G}^t, t \in \mathscr{T}_2. \end{array} \tag{7}$$

It can be observed that model (7) is easily decomposed by scenario clusters, so that it can be expressed

$$z_f^{TNF} = \sum_{c\in\mathscr{C}} z_{f,c}^{TNF}, \tag{8}$$

where

$$\begin{array}{rll} z_{f,c}^{TNF} = & \min \sum_{g\in\mathscr{G}_c} w_c^g (a^g x_c^g + b^g y_c^g) & \\ \text{s.t.} & \sum_{q\in\mathscr{A}^g} A_g^q x_c^q + B_g^q y_c^q = h^g & \forall g \in \mathscr{G}_c \\ & x_c^g = \hat{x}^g, y_c^g = \hat{y}^{g,TNF} & \forall g \in \mathscr{G}_c : t(g) \in \mathscr{T}_1 \\ & x_c^g \in \{0,1\}^{nx(g)}, y_c^g \in \mathbb{R}^{ny(g)} & \forall g \in \mathscr{G}_c : t(g) \in \mathscr{T}_2. \end{array} \tag{9}$$

3. Matheuristic algorithm H-DBFC

3.1. Dynamic candidate TNF branching

The static candidate TNF (cTNF) branching strategy used in BFC [22] consists of fixing the selected x-variable first at 0 for minimization and 1 for maximization and, afterwards, the opposite branching is considered in the already branched BF node. See other strategies in [28] for guided static branching related to the root node in the BF tree, see also [55]. By contrast, a dynamic branching strategy in DBFC is presented in this work, such that it is based on the values of the common variables, say $(x_c)_{\overline{i}}$, in the solution of the

immediate previous solving of the cluster submodels (2). So, good feasible solutions can be found earlier by the algorithm and, then, fewer cTNFs are visited during its execution. In addition to the improvement in the serial version, the scheme has significant potential in the outer parallelization version, since some simultaneous dynamic cTNF branching paths can be executed, see Section 4.

Let $(\hat{x}_c)_{\bar{i}}$ denote the value of variable $(x_c)_{\bar{i}}$ in the solution of submodels (2). Let $x_{\bar{i}}$ for $\bar{i} \in \mathscr{I}_x$ be the next branching variable in the stage-wise ordering, provided that the branching jumps to the first common variable that does not yet satisfy its NAC (3). And let $\mathscr{C}_i \subseteq \mathscr{C}$ denote the set of cluster submodels (2) that have variable x_i in common. A rough description of the *Dynamically-Guided branching* (DG) scheme for any cTNF consists of branching first on that variable in the 0-1 σ_i direction that can be expressed

$$\sigma_{\bar{i}} = \begin{cases} 0, & \text{if} \quad \sum\limits_{c \in \mathscr{C}_{\bar{i}}} (\hat{x}_c)_{\bar{i}} \leq \dfrac{1}{2}|\mathscr{C}_{\bar{i}}| \\ 1, & \text{otherwise} \end{cases}$$

Notice that, by construction, the value $\overline{x}_{\bar{i}}$ of variable $x_{\bar{i}}$ does not satisfy the NAC (3) in the solution of the submodels (2) that are used for computing $\sigma_{\bar{i}}$. Observe that $\sigma_{\bar{i}}$ is the most frequent value in the current solution of those $|\mathscr{C}_{\bar{i}}|$ submodels.

Another important feature of DBFC consists of solving the cluster submodels (2) for each x-branching path just only at the first time they are required in the BF tree and, then, they are stored for future needs. This entails a remarkable improvement in the time required for instances with a large number of quite difficult submodels at the price of a major storage effort, see Section 5.

3.2. H-DBFC strategy

The matheuristic H-DBFC for solving the original model (1) is presented in Section 3.3. The strategy is based on the relaxation of some steps of the exact DBFC algorithm and, then, the guarantee of optimality is lost. Let us point out that the breadth of the BF tree, so-called $\mathscr{BFT}$, can become an algorithmic bottleneck for instances with a large number of common variables.

The strategy performs an *incomplete backward branching* (IB) scheme, such that the previous variables in the lexicographically order are jumped back provided that they were jumped over during the forward branching because they satisfied the NAC (3). Additionally, its *stopping criterion* (SC) consists of the relative difference between two consecutive incumbent solution values being smaller than a given tolerance, say $\varepsilon > 0$. Notice that the type of *IB* branching does not allow the branching on the opposite $(1 - \sigma_i)$ direction of those variables in x_i in the already updated set $\mathscr{I}_1$, that implicitly have been branched on the 0-1 σ_i direction, by jumping over them in the forward step for choosing the next branching variable. It is pointed out that the jumping is only performed when their solution $\hat{x}_i = \sigma_i$ satisfies their NAC in the submodels (2). Alternatively, different types of stronger strategies could be considered for solving problems with smaller dimensions that those whose results are reported in section 5. Examples of those strategies are: i) keeping backward branching on reverting to the cTNF with the immediate previous common variable in the lexicographic order or ii) skipping the solving of the scenario cluster submodels (9) at the integer TNFs.

3.3. Algorithm H-DBFC

The H-DBFC proposal is formally presented in Algorithm 1. Let us describe the main steps of the algorithm.

The input parameter κ_{max} is a modeler-driven limit on the number of times that the models (6) and (9) are allowed to be solved (in Sep 5) for any integer TNF; and v is a counter of the number of incumbent feasible solutions found along the iterations, so that it plays the role of the index for the related incumbent value, $\overline{z}_v^{DEM}$.

In Step 1 a scenario cluster based lower bound, say $\underline{z}^{DEM}$, is obtained by solving the MIP submodels (2) for all clusters. Additionally, parameter σ_i, $i \in \mathscr{I}_x$ is computed for future needs while selecting in H-DBFC the 0-1 branching direction on the next branching (common) variable.

The stage-wise ordering of the variables allows the branching on a common variable to be selected in Step 2 for the given cTNF, instead of considering a fixed selection for the BF node to be branched on (as done in BFC). Additionally, the selection of the branching variable $x_{\bar{i}}$ can be dynamically replaced in some situations (see below) by jumping to the first common variable does not yet satisfy the NAC (3) in the last time that the cluster submodels (2) are solved.

The Dynamically-Guided (DG) branching scheme in Step 3 of H-DBFC requires to recover the solution of the common variables from the immediate previous solving of the scenario cluster submodels (2), so that the 0-1 direction of the branching variable $x_{\bar{i}}$ is selected based on the 0-1 σ-parameter, see Section 3.1, as opposed to the static branching criterion used in BFC.

The branching forward scheme used in Step 2 and Step 3 of H-DBFC has a significantly simpler encoding than the one in BFC (Step 2 to Step 5). Notice that the stage-node variable control in BFC is no longer needed. Instead, it suffices to consider the frequency of the 0-1 values of the common variables in the stage-wise ordering, since the variable branching direction is chosen based on the $\sigma-$ parameter.

In Step 4 the cluster submodels (2) are solved for a given TNF that has been built with the branching path included by the related 0-1 σ-directions of a subset of common variables (whose indices are in set $\mathscr{I}_1$) up to the last branched variable $x_{\bar{i}}$ in the 0-1 $\sigma_{\bar{i}}$ direction. Next, the forward step jumps from the current branching variable to the first one in the lexicographical order of set $\mathscr{I}_x \setminus \mathscr{I}_1$ whose value in the solution of the submodels (2) does not satisfy the NAC (3). This very important feature does not exists in BFC and it is crucial for speeding up the convergence of the algorithm.

Step 5 for the integer TNF includes a set of submodels to be solved. It is worth to point out that model (6) for $\kappa > 1$ consists of the values of the solution of model (9) for the x-variables related to the nodes at stages $t > t^*$ in the scenario tree. Notice that for both models the values of the x-variables in stages up to t^* are the ones obtained in Step 4.

Step 4 and 5 include the stopping criterion based on the new feasible solution that is found.

Step 7 performs the backward branching scheme, branching back to the variable according to the explicitly branched forward criterion.

Notice that the pruning check in Step 8 and the opposite branching direction in Step 9 are performed according to the $\sigma-$ parameter, so, the static branching criterion used in BFC is not longer used in DBFC.

The matheuristic character of H-DBFC is due to the scheme *IB* as well as the stopping criterion to start with. Additionally, notice that the values of the *y*-variables for the nodes in the stages up to the break one in model (6) to be solved at any integer TNF (Step 5) are *only* obtained for the 0-1 values of the *x*-variables in the solution of the cluster submodels (2) that are solved in the Step 4. Observe also that an additional scheme such as the one presented in Step 7 of BFC [22] could be used to guarantee optimality. However, given the large size of the instances that are aimed with H-DBFC, see Section 5, the computing effort would be unaffordable and, thus, preventing such scheme from being used in the approach.

Remark 3. *The selection of the break stage is crucial for the efficiency of the algorithm. The smaller the break stage t^*, the smaller the cardinality of $\mathscr{I}_x$. And then, the depth of the BF trees can likely be smaller as well as the higher the chance in Step 4 of the matheuristic algorithm H-DBFC (presented next) that the solution of the decomposition cluster submodels* (2) *can satisfy the NAC* (3) *of the common variables. So, the convergence for obtaining feasible solutions in model* (6) *and submodels* (9) *is accelerated. However, the smaller the break stage t^*, the higher the dimensions of the submodels* (2) *and* (9)*. So, an instance-driven balance is required.*

Algorithm 1 H-DBFC algorithm

Step 0: **(Initialization)**
Set $\nu := 0$, $\overline{z}_\nu^{DEM} := \infty$, $\overline{i} := 0$, $\mathscr{I}_1 := \{\overline{i}\}$.

Step 1: **(Root TNF)**
Solve the scenario cluster MIP submodels (2) to obtain $z_c,\ \forall c \in \mathscr{C}$.
Compute $\underline{z}^{DEM} = \sum_{c\in\mathscr{C}} z_c$ (lower bound) and $\sigma_i,\ \forall i \in \mathscr{I}_x$.
If any $(x_c)_i$ variable does not satisfy NAC (3), $i \in \mathscr{I}_x, c \in \mathscr{C}$, then go to Step 2.
If any $(y_c)_i$ variable does not satisfy NAC (4), $i \in \mathscr{I}_y, c \in \mathscr{C}$, then go to Step 5.
Otherwise, $\nu := \nu + 1$, $\overline{z}_\nu^{DEM} := z$ and go to Step 10.

Step 2: **(Forward TNF branching)**
Reset $\overline{i} := \overline{i} + 1$. Update $\overline{i}$ according to the first variable (in the lexicographic order) that does not satisfy NAC.
Update $\mathscr{I}_1 := \mathscr{I}_1 \cup \{\overline{i}\}$.

Step 3: **(Dynamically Guided branching)**
Set $(x_c)_{\overline{i}} := \sigma_{\overline{i}},\ \forall c \in \mathscr{C}_{\overline{i}}$.

Step 4: **(Candidate TNF)**
Solve the scenario cluster MIP submodels (2) to obtain $z_c,\ \forall c \in \mathscr{C}$.
Compute $z = \sum_{c\in\mathscr{C}} z_c$ and $\sigma_i,\ \forall i \in \mathscr{I}_x \setminus \mathscr{I}_1$.
If $z \geq \overline{z}_\nu^{DEM}$, then go to Step 6.
If any variable $(x_c)_i$ does not satisfy NAC (3) $i \in \mathscr{I}_x, c \in \mathscr{C}$, then go to Step 2.
If any variable $(y_c)_i$ does not satisfy NAC (4) $i \in \mathscr{I}_y, c \in \mathscr{C}$, then go to Step 5.
Update $\nu := \nu + 1$, $\overline{z}_\nu^{DEM} := z$.
Test the stopping criterion: if $\left|\frac{\overline{z}_{\nu-1}^{DEM} - \overline{z}_\nu^{DEM}}{\overline{z}_\nu^{DEM}}\right| < \varepsilon$, then go to Step 10.
Go to Step 6.

Step 5: **(Integer TNF models)**
Reset $\kappa := 1$.

Step 5.1: Solve LP model (6) to obtain z_{LP}^{TNF}.
If it is feasible and $z_{LP}^{TNF} < \overline{z}_\nu^{DEM}$, then update $\nu := \nu + 1$, $\overline{z}_\nu^{DEM} := z_{LP}^{TNF}$.
If $\kappa = \kappa_{max}$, then if $\left|\frac{\overline{z}_{\nu-1}^{DEM} - \overline{z}_\nu^{DEM}}{\overline{z}_\nu^{DEM}}\right| < \varepsilon$, then go to Step 10.
Solve the submodels (9) to obtain $z_{fc}^{TNF},\ \forall c \in \mathscr{C}$.
If all of them are feasible, then compute $z_f^{TNF} = \sum_{c\in\mathscr{C}} z_{fc}^{TNF}$ and
if $z_f^{TNF} < \overline{z}_\nu^{DEM}$, then update $\nu := \nu + 1$, $\overline{z}_\nu^{DEM} := z_f^{TNF}$.
If all the x variables from (9) are the same as in the (6) solution, then $\kappa := \kappa_{max}$.
If $\kappa = \kappa_{max}$, then if $\left|\frac{\overline{z}_{\nu-1}^{DEM} - \overline{z}_\nu^{DEM}}{\overline{z}_\nu^{DEM}}\right| < \varepsilon$, then go to Step 10.
If $\kappa < \kappa_{max}$, then update $\kappa := \kappa + 1$ and go to Step 5.1.

Step 6: **(Branch pruning)**.
If $(x_c)_{\overline{i}}$ has been branched on to $\sigma_{\overline{i}}$ for any $c \in \mathscr{C}_{\overline{i}}$, then go to Step 9.

Step 7: **(Backward TNF branching)**
Reset $\overline{i} := \overline{i} - 1$. Update backward $\overline{i}$ according to the branching forward scheme.
If $\overline{i} = 0$, then go to Step 10.

Step 8: **(Prune checking)**
If $(x_c)_{\overline{i}} = 1 - \sigma_{\overline{i}}$ for any $c \in \mathscr{C}_{\overline{i}}$, then go to Step 7.

Step 9: **(Opposite branching)**
Reset $(x_c)_{\overline{i}} := 1 - \sigma_{\overline{i}},\ \forall c \in \mathscr{C}_{\overline{i}}$ and go to Step 4.

Step 10: **(End of the algorithm)**
The H-DBFC solution, its value $\overline{z}_\nu^{DEM}$ and $OG = \frac{\overline{z}^{DEM} - \underline{z}^{DEM}}{\underline{z}^{DEM}}$% have been found, **STOP**.

4. Outer and Inner parallelization of H-DBFC

Large sized multistage stochastic mixed 0-1 optimization problems are very difficult to solve, mainly due to the number of constraints and the 0-1 variables. In order to improve the performance of our matheuristic H-DBFC algorithm for solving large sized problems, Section 4.1 presents some considerations based on the performance of the parallel version presented in [4] for BFC. On one hand, the so-called inner parallelization reduces the execution time for solving the independent submodels that are associated with each TNF. On the other hand, the so-called outer parallelization reduces the number of TNFs to be visited by splitting the BF tree and, then, allowing to obtain earlier tighter bounds.

Parallel algorithms have been developed in stochastic optimization for linear programs, see [12, 54] as we know for the first parallel versions of Benders decomposition; see [9, 19] for two-stage programs and [4, 13, 47, 48, 55, 61, 64, 70, 74] for multistage ones. For some applications in several fields, see [18, 20, 27, 43, 44]. Recently, [55] has presented a parallelization of the algorithm BFC. In this section the inner and outer parallel versions of H-DBFC are presented.

4.1. Parallelization schemes

We use the following definitions of the three thread types given in [4] for parallelizing H-DBFC (see Algorithm 1): *coordinator thread*, where the non-parallelized tasks are executed; *primary thread*, solves the scenario cluster submodels (2) and (9) in parallel; and *auxiliary thread*, used by each call to the MIP solver of choice. Then, denote as $(a \times b \times h)$ the thread configuration of a joint outer-inner parallelization execution, where a is the number of coordinator threads (one for each *path*, see below), b is the number of primary threads associated with each coordinator thread (including itself) and h is the number of auxiliary threads associated with each primary thread (including itself). If there are 64 threads available, as an example, configuration $(2 \times 4 \times 8)$ means that 2 coordinator threads are defined (associated with 2 paths in the outer parallelization scheme, see below), 4 threads for solving the scenario cluster submodels (2) and (9) in parallel (associated with the inner parallelization scheme) and 8 threads to be used by the MIP solver of choice.

The inner parallelization in H-DBFC has the configuration $(1 \times b \times h)$ where $b \leq C$, such that the parallelization is performed for solving up to the C cluster submodels (2) in Step 1 and Step 4 as well as to solve the C submodels (9) in Step 5. Different strategies can be used depending on the number of *primary threads* to use in distributed memory (MPI threads) and the number of *auxiliary threads* to use in shared memory (by the MIP solver of choice). As an example, assume that 8 threads are available for parallelizing the C independent submodels. Some options are as follows: Configuration $(1 \times 8 \times 1)$, 8 MPI threads are used for solving the submodels (one per each thread) and running the MIP solver in only one auxiliary thread for each primary thread (i.e., the MIP solver is not allowed to use internal functions in parallel); $(1 \times 4 \times 2)$, 4 MPI threads are used with 2 MIP solver threads each; $(1 \times 2 \times 4)$, 2 MPI threads are used with 4 MIP solver threads each; and $(1 \times 1 \times 8)$, 1 MPI thread and 8 MIP solver threads. Elsewhere [4] we present a wider description of the inner parallelization.

The outer parallelization is managed by the so-called paths. Assume p coordinator threads, the procedure starts by defining a set of 0-1 variables in a lexicographical form, such that the combinations of their 0-1 values allow the p paths to be initiated. Each combination of the 0-1 values of those variables is implemented by a coordinator thread and it is associated with the following elements: (1) Dynamically reassigned subproblem of an unvisited BF tree, defined by a path BF tree, say $\mathscr{BFT}_{path}$, and its corresponding root TNF, say $\hat{\mathscr{N}}_{path} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k)$; (2) Algorithm H-DBFC; (3) Set of primary and auxiliary threads; and (4) MPI (Message Passing Interface) environment with other coordinator threads. Its efficiency is greatly increased by the interconnect synchronization of the executions of H-DBFC. It is obtained by using the following elements: (a) Local incumbent solution values, say $\overline{z}_{path}^{DEM}$, of the subproblems handled by the coordinator threads, making earlier branch pruning, and (b) The reassignment of the path subproblem in those threads associated with paths that have no more TNF to visit, by redefining the root TNF, $\hat{\mathscr{N}}_{path}$ and the related path

BF tree, $\mathscr{BFT}_{path}$. This is the result of splitting the tree of an unfinished $\mathscr{BFT}$ path. So, the parallelization of the set of paths may shorten the number of TNFs to be visited and the reduction of the computing time, such that the upper bound of the solution of the original model (1) can be tightened.

The outer parallelization of H-DBFC has the following steps:

- Starting the parallel executing of the p paths, each one with its root TNF $\hat{\mathscr{N}}_{path}$. The optimization and MPI variables are declared and the global environment is defined for all coordinator threads.
- The execution of each path consists of executing the steps of H-DBFC from its root path until the starting of the synchronization phase (see Algorithm 2).
- A path reaches the synchronization phase, illustrated with a dashed red box in Figure 1, when it satisfies one of the following conditions, at least: (i) all BF nodes have been visited, and (ii) an integer TNF have just been visited. Table 1 illustrates such situations in an example. It is important to underline that the dynamically-guided (DG) branching in H-DBFC aims precisely to visit integer TNFs often, therefore, the idle time is reduced with respect to the strategy that we have presented elsewhere, see [4].
- The synchronization phase is activated once all paths have reached it. See next how it operates.

The operation of the synchronization phase is as follows: First, all paths gather the current incumbent solution value obtained by each path $\overline{z}^{DEM}_{path}$ and update the incumbent global path $\overline{z}^{DEM}$ of the original model (1). Second, the dead/active path analysis starts and one of the following three situations arises:

- If all paths are dead, then all TNFs of the BF trees $\mathscr{BFT}_{path}$ have been visited, so, the original BF tree $\mathscr{BFT}$ have been fully visited and the execution end in all paths, so, the outer parallel version of H-DBFC stops (global end).
- If all paths are active, then each one will continue branching in its own $\mathscr{BFT}_{path}$ with the new updated $\overline{z}^{DEM}$.
- Otherwise, each dead path will match with an active path. The outer parallelization presented here has been extended to deal with the dynamic and matheuristic nature of the H-DBFC algorithm, as opposed to the root node matching criterion in [4]. Thus, only the explicitly branched forward variables are candidates for the matching, since their influence is considered to be more significant. Let us $path_1$ and $path_0$ be a dead and an active path, respectively. Consider that the down levels l_1 and l_0 have been explicitly reached in the BF tree from the root node, respectively, i.e., the root TNF for current active $path_0$ in the matheuristics perspective is $\hat{\mathscr{N}}_{path_0} = (\hat{x}^0_1, \ldots, \hat{x}^0_{l_0})$, and the root TNF for the current dead $path_1$ is $\hat{\mathscr{N}}_{path_1} = (\hat{x}^1_1, \ldots, \hat{x}^1_{l_1})$, such that the situation is as follows:
  - All the successor TNFs from the dead path have already been pruned;
  - The dead path $path_1$ will restart the execution of H-DBFC from the root TNF $\hat{\mathscr{N}}_{path_1} = (x_1, \ldots, x_{l_0}, x_{l_0+1}) = (\hat{x}^0_1, \ldots, \hat{x}^0_{l_0}, 1-\sigma_{l_0+1})$ and its associated new $\mathscr{BFT}_{path_1}$.
  - The new starting combination of 0-1 values in $\mathscr{BFT}_{path_1}$ is included by values of the variables already branched on or fixed at in the current branching of $path_0$ plus the opposite value for the new (next) explicitly branched variable with respect to $path_0$ branching. See Section 3.1 for the description of the σ-parameter. Notice that in [4] the dead path is always branched to value 1 in the matching, due to the static branching criterion.
  - The active $path_0$ will continue branching as stated in H-DBFC, but its root TNF has been updated to $\hat{\mathscr{N}}_{path_0} = (x_1, \ldots, x_{l_0}, x_{l_0+1}) = (\hat{x}^0_1, \ldots, \hat{x}^0_{l_0}, \sigma_{l_0+1})$.

Algorithm 2 Synchronization phase

(All paths gather $\overline{z}_{path}^{DEM}$ and $dead_{path}$ status) [All coordinator threads].
When a path is dead or has obtained a feasible solution,
set $\overline{z}^{DEM} = min(\overline{z}^{DEM}, min_{path}\{\overline{z}_{path}^{DEM}\})$.
If all $dead_{path} = 1$ (dead) then
    (Finish MPI environment) [All coordinator threads].
    GLOBAL END.
else
    (Dead paths receive active paths variable branching)
    [All coordinator threads].
    Dead paths are reassigned by splitting active path $\mathscr{BFT}_{path}$.
    All paths update root TNF $\hat{\mathscr{N}}_{path}$.
Paths where $dead_{path} = 0$ (active) continue branching.
Paths where $dead_{path} = 1$ restart H-DBFC at the partial branching of
its related matched path : $dead_{path} = 0$, see Algorithm 1.

4.2. Illustrative example

Figure 2 and Table 1 show an illustrative example of the outer parallel version of H-DBFC while solving P3, a randomly generated instance taken from [4, 22]. It uses two paths, where the TNFs of Path 1 and Path 2 are in blue and green, respectively. Notice that when two paths are available only one 0-1 variable can be fixed for the initial definition of the path, say variable x_1. Thus, Path 1 starts the algorithm with root TNF $\hat{\mathscr{N}}_1 : (x_1) = (0)$ and Path 2 with $\hat{\mathscr{N}}_2 : (x_1) = (1)$. The synchronization phase is reached four times during the global solving, see below for the process.

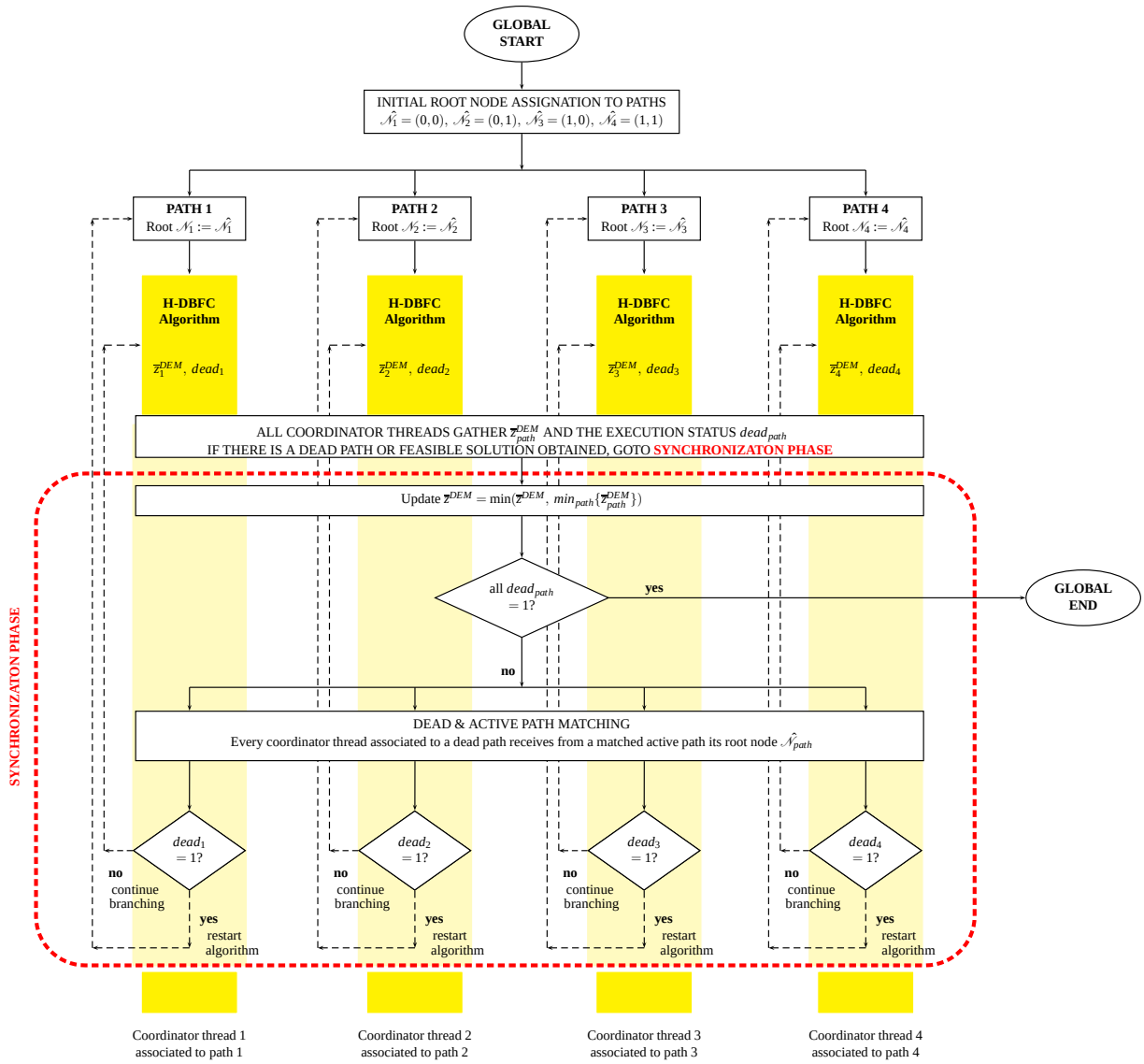

Figure 1: Illustrative 4-path Outer PH-DBFC diagram, $(4 \times 1 \times 1)$

The first synchronization phase, denoted *Sync*.1 in Figure 2, takes place when Path 1 obtains a feasible solution (for the original model (1), that is, all common variables take 0-1 values) at TNF 06, where $\bar{z}_1^{DEM} = -291665$, and as it is still active, so $dead_1 := 0$; while Path 2 obtains a feasible solution at TNF 01, $\bar{z}_2^{DEM} = -290398$, so, the branch is pruned and, then, $\mathscr{BFT}_2$ is fully visited, so, $dead_2 := 1$. After the comparison of the path solution values, as $\bar{z}_1^{DEM} < \bar{z}_2^{DEM}$ and $\bar{z}_1^{DEM} < \bar{z}^{DEM}$, then $\bar{z}^{DEM} := \bar{z}_1^{DEM}$. As Path 2 is dead, let us descend to branch on a common variable, say x_2 by sharing Path 1 BF tree. The root TNF for Path 1 is updated to $\hat{\mathcal{N}}_1 : \ (x_1, x_2) = (0,0)$ and continue branching. On the other hand, Path 2 will give up the previous BF tree and, being linked to the root TNF $\hat{\mathcal{N}}_2 : \ (x_1, x_2) = (0,1)$, it will restart H-DBFC to solve the new path BF tree $\mathscr{BFT}_2$. Notice that root TNFs are indicated with double lined circles in Figure 2.

The second synchronization phase, denoted *Sync*.2, takes place when Path 1 obtains a feasible solution at TNF 07 (whose value is smaller than the global value $\bar{z}^{DEM}$) where $\bar{z}_1^{DEM} = -291709$ and it is still active.

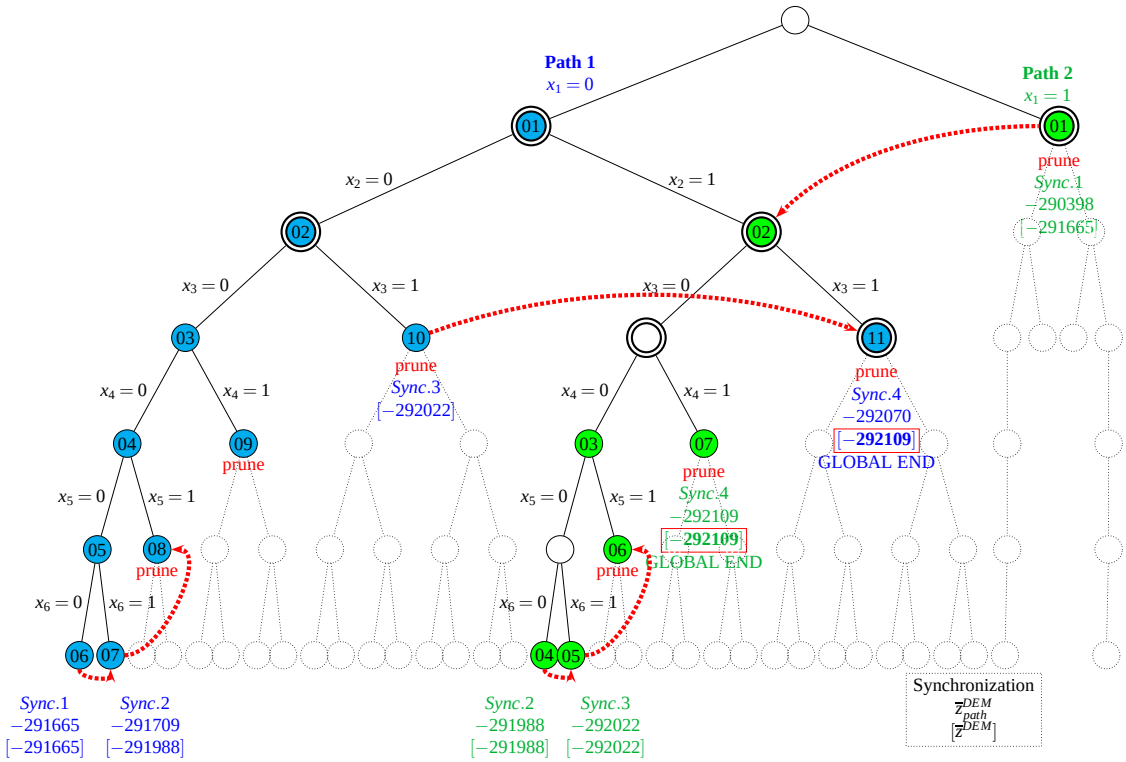

Figure 2: Two-path Outer P-BFC performance for instance P3

On the other hand, Path 2 branches on TNFs 02 to 04, such that a feasible solution is obtained, where $\overline{z}_2^{DEM} = -291988$ (smaller than the global incumbent value) and it is still active. After the comparison of the path solution values, since $\overline{z}_1^{DEM} > \overline{z}_2^{DEM}$ and $\overline{z}_2^{DEM} < \overline{z}^{DEM}$, it results that $\overline{z}^{DEM} := \overline{z}_2^{DEM}$, and TNFs 07 and 04 are pruned in Path 1 and Path 2, respectively. As $dead_1 := 0$ and $dead_2 := 0$, root TNFs do not need to be updated and both paths continue branching.

The third synchronization phase, denoted *Sync*.3, takes place when Path 1 branches on TNFs 08 to 10 and, finally, it stops since it has finished branching at its own tree rooted at $(0,0)$. On the other hand, Path 2 branches on TNF 05 where a feasible solution is obtained, where $\overline{z}_2^{DEM} = -292022$ (smaller than the global incumbent value). After the comparison of the path solution values, as $\overline{z}_1^{DEM} > \overline{z}_2^{DEM}$ and $\overline{z}_2^{DEM} < \overline{z}^{DEM}$, then $\overline{z}^{DEM} := \overline{z}_2^{DEM}$. TNF 10 of Path 1 is pruned and the path is dead, $dead_1 := 1$, since Path 2 is still active, $dead_2 := 0$. Path 1 has finished its own BF tree, therefore, let us descend to branch a common variable, say x_3. Path 1 will give up the previous BF tree and, being linked to root TNF $\hat{\mathcal{N}}_1 : (x_1, x_2, x_3) = (0,1,1)$, the root TNF for Path 2 is updated to $\hat{\mathcal{N}}_2 : (x_1, x_2, x_3) = (0,1,0)$.

Finally, the last synchronization phase, denoted *Sync*.4, takes place when Path 1 branches on TNF 11, a feasible solution is obtained, where $\overline{z}_1^{DEM} = -292070$ (smaller than the global incumbent solution). On the other hand, Path 2 branches on TNFs 06 and 07, where a feasible solution is obtained, where $\overline{z}_2^{DEM} = -292109$ (smaller than the global incumbent value). After the comparison of the path solution values, as $\overline{z}_1^{DEM} > \overline{z}_2^{DEM}$ and $\overline{z}_2^{DEM} < \overline{z}^{DEM}$, then $\overline{z}^{DEM} := \overline{z}_2^{DEM}$. TNF 11 of Path 1 is pruned. And, additionally, both paths are dead, $dead_1 := 1$ and $dead_2 := 1$, since all TNFs at the path trees $\mathscr{BF}\mathscr{T}_1$ and $\mathscr{BF}\mathscr{T}_2$ have been branched on. So, the outer parallel version of H-DBFC has finished, and the incumbent solution value is $\overline{z}^{DEM} = -292109$.

Path 1			Path 2		
Root/TNF	z_1^c	$\overline{z}_1^{DEM}$	Root/TNF	z_2^c	$\overline{z}_2^{DEM}$
Root (0)			Root (1)		
TNF 01	-292117		TNF 01	-290404	-290398
TNF 02	-291839				
TNF 03	-291826				
TNF 04	-291759				
TNF 05	-291741				
TNF 06	-291678	-291665			
Obtain feas soln, $dead_1 = 0$			Obtain feas soln and all TNFs branched, $dead_2 = 1$		
Synchronization phase 1: $\overline{z}^{DEM} = -291665$					
Continue branching in H-DBFC			Restart H-DBFC		
Root $(0,0)$			Root $(0,1)$		
TNF 07	-291718	-291709	TNF 02	-292116	
			TNF 03	-292061	
			TNF 04	-291996	-291988
Obtain feas soln, $dead_1 = 0$			Obtain feas soln, $dead_2 = 0$		
Synchronization phase 2: $\overline{z}^{DEM} = -291988$					
Continue branching in H-DBFC			Continue branching in H-DBFC		
Root $(0,0)$			Root $(0,1)$		
TNF 08	-291652		TNF 05	-292027	-292022
TNF 09	-291826				
TNF 10	-291803	∞			
All TNFs branched, $dead_1 = 1$			Obtain feas soln, $dead_2 = 0$		
Synchronization phase 3: $\overline{z}^{DEM} = -292022$					
Restart H-DBFC			Continue branching in H-DBFC		
Root $(0,1,1)$			Root $(0,1,0)$		
TNF 11	-292075	-292070	TNF 06	-291733	
			TNF 07	-292114	-292109
Obtain feas soln and all TNFs branched, $dead_1 = 1$			Obtain feas soln and all TNFs branched, $dead_2 = 1$		
Synchronization phase 4: $\overline{z}^{DEM} = -292109$					
GLOBAL END					

Table 1: Two-path Outer PH-DBFC performance

5. Computational experience

The computational experiments were conducted in the ARINA computational cluster from SGI/IZO-SGIker at UPV/EHU, which provides 1926 cores divided as follows: 1774 xeon cores, 248 Itanium2 cores and 40 opteron cores. All computing nodes are connected by an Infiniband network with high bandwidth and low latency. For this computational experiment xeon x86_64 architecture (Xeon Nehalem-EP E5520 @ 2.4GHz) type nodes were used, consisting of 18 nodes, such that each node (with 2 processors of 6 threads) has 48 Gb of RAM, 2.4 Ghz and a QDR infiniband interconnection. A 22 Tb high performance file system based on Lustre was used for data storage.

The inner and outer parallel versions of matheuristic H-DBFC were implemented in a C++ experimental code which uses the state-of-the-art optimization LP/MIP solver CPLEX v12.5 [38] called from the open source library COIN-OR v1.3.1 [15]. The optimizer is used to solve the MIP submodels (2) for the set of scenario clusters $\mathscr{C}$ in Step 1 and Step 4, and the LP submodel (6) and the MIP submodels (9) for the set of scenario clusters $\mathscr{C}$ in Step 5.

The computational experience is reported as follows: Section 5.1 presents the dimensions of the instances we have experimented with. Section 5.2 details the performance of the serial version of the matheuristic H-DBFC, and Sections 5.3 and 5.4 detail the performance of the parallel versions of H-DBFC, so-called inner IH-DBFC and outer OH-DBFC, respectively.

5.1. Testbed dimensions

We have considered in our computational experience 18 large-sized instances from a realistic production planning problem taken from [3, 16], whose dimensions are shown in Table 2. The headings are as follows: T, number of stages; m, number of constraints; nx, number of 0-1 variables; nxc, number of common variables, i.e., number of 0-1 variables in the subset of stages up to t^*; ny, number of continuous variables; nel, number of non zero coefficients in the constraint matrix; $dens$, constraint matrix density (in %); $|\Omega|$, number of scenarios; $|\mathscr{G}|$, number of nodes in the scenario tree; and $Tree$, scenario tree structure $A_1^{B_1}A_2^{B_2}A_3^{B_3}$, where A_i denotes the number of children that each node has in each stage of block i, and B_i denotes the number of stages of block i, for $i = 1,2,3$. Note: The break stage has been equal to 1 for all experiments that are reported, but for instance c47 where $t^* = 2$ in Tables 3 and 4, and c55 where $t^* = 6$ in Tables 3 and 4 and $t^* = 3$ in Table 5. All results reported in the section consider that the ε-stopping parameter is 0.005.

Table 2: Original model (1) dimensions (compact representation)

Instance	T	m	nx	nxc	ny	nel	$dens$	$\|\Omega\|$	$\|\mathscr{G}\|$	$Tree$
c43	8	163080	42750	300	106650	892975	0.0037	432	855	$1^6 3^3 2^4$
c44	8	167355	42750	300	106650	1072525	0.0043	432	855	$1^6 3^3 2^4$
c45	8	234660	66000	600	154800	1635440	0.0032	432	660	$1^6 2^4 3^3$
c46	8	241260	66000	600	154800	2142320	0.0040	432	660	$1^6 2^4 3^3$
c47	8	316755	85500	600	213300	2134790	0.0023	432	855	$1^6 3^3 2^4$
c48	8	325305	85500	600	213300	2798270	0.0029	432	855	$1^6 3^3 2^4$
c49	9	71148	19560	60	45720	200966	0.0043	1296	1956	$1^6 2^4 3^4$
c50	9	75060	19560	60	45720	226394	0.0046	1296	1956	$1^6 2^4 3^4$
c51	9	96948	25560	60	63720	262898	0.0030	1296	2556	$1^6 3^4 2^4$
c52	9	102060	25560	60	63720	311462	0.0034	1296	2556	$1^6 3^4 2^4$
c53	11	392436	103320	60	258120	1115486	0.0008	5184	10332	$1^6 3^4 2^6$
c54	11	413100	103320	60	258120	1239470	0.0008	5184	10332	$1^6 3^4 2^6$
c55	11	448020	116840	60	272760	1340022	0.0008	7776	11684	$1^6 2^5 3^5$
c56	11	424652	116840	60	272760	1199814	0.0007	7776	11684	$1^6 2^5 3^5$
c57	9	693876	195600	600	457200	4775444	0.0011	1296	1956	$1^6 2^4 3^4$
c58	9	732996	195600	600	457200	7903088	0.0017	1296	1956	$1^6 2^4 3^4$
c59	9	946476	255600	600	637200	6411860	0.0008	1296	2556	$1^6 3^4 2^4$
c79	8	3032055	769500	5400	1919700	253365120	0.0031	432	855	$1^6 3^3 2^4$

Observe the high number of stages and scenarios, and the high dimensions of the largest instance, c79. It is interesting to point out that the dimension is not the only significative factor for determining computational complexity. The instances have a very nice structure, under a symmetric tree, but the behaviour can be very different.

5.2. Performance of serial matheuristic SH-DBFC

Table 3 shows the performance of SH-DBFC with respect to our Stochastic Dynamic Programming-based matheuristic serial version, S-SDP, as presented in [3] and the plain use of CPLEX, with thread configuration $(1 \times 1 \times 8)$. The headings are as follows: *Inst*, instance's code; z^{DEM}, $\overline{z}^{DEM}_{S-SDP}$ and $\overline{z}^{DEM}_{SH-DBFC}$, solution value of the original model (1) and incumbent values of S-SDP and SH-DBFC, respectively; *OG*, optimality gap in percentage shown by CPLEX, that is, the relative difference between the incumbent solution value and the value of the objective function of the best active *B&B* node; *GG*% goodness gap, relative difference (in percentage) between the solution values provided by the matheuristics S-SDP and SH-DBFC, and the CPLEX

incumbent solution value, whose expression is $GG\% = 100 \cdot \frac{\overline{z}^{DEM}_{(.)} - z^{DEM}}{z^{DEM}}$, where $(.)$ is S-SDP or SH-DBFC; $ti(s)$, time instant (in seconds) at which the CPLEX incumbent solution is found; and $t(s)$, total elapsed time (in seconds).

The results are only reported for the matheuristic strategy with the most accurate break stage t^*, as mentioned above. The strategy significantly reduces the number of candidate TNFs and integer TNFs visited (and, thus, the number of cluster submodels (2) and (9) solved is significantly small); therefore, the elapsed time required is also reduced.

Table 3: SH-DBFC performance versus S-SDP and plain use of CPLEX

	CPLEX $(1\times1\times8)$				S-SDP $(1\times1\times1)$			SH-DBFC $(1\times1\times1)$			
Inst	z^{DEM}	$OG\%$	$ti(s)$	$t(s)$	$\overline{z}^{DEM}_{S-SDP}$	$GG\%$	$t(s)$	t^*	$\overline{z}^{DEM}_{SH-DBFC}$	$GG\%$	$t(s)$
c43	3498249 [b]	0.06	4343	16212	3539594	1.18	**33**	1	3505629	**0.21**	161
c44	4211366 [b]	0.03	161	24252	4249979	0.92	145	1	4226605	**0.36**	**86**
c45	8036004	*	27	37	8123167	**1.08**	**28**	1	8127924	1.14	201
c46	8087808	*	74	375	8139108	**0.63**	**51**	1	8149793	0.77	186
c47	7151251 [b]	0.09	240	9421	7227178	1.06	433	2	7175002	**0.33**	**227**
c48	6594167 [b]	0.12	363	9007	6660629	1.01	**108**	1	6603791	**0.15**	363
c49	993334	*	1	1	1004692	1.14	23	1	993433	*****	**1**
c50	1005119	*	2	2	1006477	0.14	4	1	1005119	*****	**3**
c51	772567	*	16	16	775933	0.44	14	1	772590	*****	14
c52	862754	*	20	20	870090	0.85	18	1	863566	**0.09**	**15**
c53	670234 [b]	0.32	1381	15091	685613	2.29	**59**	1	675116	**0.73**	371
c54	769236 [b]	0.19	2395	16938	776389	0.93	486	1	774450	**0.68**	**183**
c55	1163290 [b]	0.07	6967	8948	1165132	**0.16**	73	6	1169030	0.49	**28**
c56	1126270	*	43	43	1128968	**0.24**	**20**	1	1129264	0.27	37
c57	7174215 [b]	0.06	3829	3829	7256183	1.14	**91**	1	7233506	**0.83**	299
c58	8753936 [b]	0.02	22629	22629	8803020	**0.56**	**200**	1	8863723	1.25	212
c59	8200795 [b]	0.08	2094	15974	8251017	**0.61**	**171**	1	8253812	0.65	192
c79	$-$ [a]	$-$	8h	8h	61360087	$-$	**14473**	1	61118713	-	14511
Average	4062976	0.06	2623	8400	4097833	0.85	**115**	1	4089550	**0.47**	152

*: Optimality/goodness gap achieved $(< 0.01\%)$
$-$: non available
[a]: Time limit (8h) exceeded
[b]: Out of memory (35 Gb)

Notice that the choice of the break stage t^* is very much instance dependent. Observe the results obtained by the three approaches for instances c55 and c56, whose *Tree* structure and dimensions are identical, but the realizations of the scenarios are different. CPLEX provides similar solution value for both instances, but the elapsed time is 8948 seconds for c55 and 43 seconds for c56. The matheuristic algorithms provide solutions whose GG are similar for the same instance with impressive elapsed times, but SH-DBFC requires that t^* is changed from 1 to 6 for instance c55. In any case, it is clear that the time complexity depends not only on the model dimensions but also on its tightness.

The last row of Table 3 (as it happens for Tables 4 and 5) reports the average value for each column, where the largest instance, c79, has not been included in the computation, rather it is separately considered.

Observe in Table 3 that the quality of the solution value is practically the same in the instances c49-c52 for CPLEX and SH-DBFC, and the solution value provided by SH-DBFC is better in two thirds of the instances

(in fact, 67%) than the one provided by S-SDP. The reduction in the elapsed time is remarkable in comparison with plain use of CPLEX in both matheuristics, even with respect to the time instant at which the best known CPLEX solution is found. Notice that the instances are generally solved in very few minutes, being c47, c54 and c79 the hardest ones for S-SDP and c48, c53 and c79 the hardest ones for SH-DBFC. Observe that 7 out of the 18 instances are solved by SH-DBFC in less time than S-SDP. The average GG for instances c43 to c59 in SH-DBFC is 0.47% versus 0.85% in S-SDP, while the average elapsed times are 152 and 115 seconds, respectively. For the biggest instance c79, where plain use of CPLEX can not even obtain the LP solution value in the 8h time limit, both matheuristics require similar times (4h approx). On the other hand, SH-DBFC obtains a better solution value than S-SDP in a relative difference of 0.39%. The break stage selection is an important decision, since the smaller its value, the tighter the TNF bounds but more elapsed time is required by the submodels solving. The appropiate selection depends on the problem dimensions and its complexity.

See in Appendix A some figures about the evolution of the performance of the solution values obtained by H-DBFC and SDP.

5.3. Performance of inner parallelization IH-DBFC

Table 4 shows the main results of the inner parallel IH-DBFC, thread configuration ($1 \times 2 \times 1$), where 2 is the number of primary threads. It is also shown its comparison with the inner parallel version, IP-SDP, of matheuristic SDP [3], thread configuration ($1 \times 12 \times 1$). The new headings are as follows: $\overline{z}^{DEM}_{IP-SDP}$ and $\overline{z}^{DEM}_{IH-DBFC}$, incumbent solutions of IP-SDP and IH-DBFC, respectively; $GG\%$ goodness gap, the relative difference (in percentage) between the solution values provided by the inner parallel matheuristics IP-SDP and IH-DBFC, and the CPLEX incumbent solution value, whose expression is $GG\% = 100 \cdot \frac{\overline{z}^{DEM}_{(.)} - z^{DEM}}{z^{DEM}}$, where $(.)$ is IP-SDP or IH-DBFC; C, number of scenario clusters (i.e., number of scenario tree nodes for the stage t^*+1); $\overline{c}$, average number of scenario cluster submodels (2) by iteration that are solved in Step 4 (candidate TNF branching) of the inner parallel version of DBFC; $S_{th} = t_{serial}/t_{inner}$, speed up when using th primary threads; $E_{th} = 100 \cdot S_{th}/th$, efficiency when using th primary threads.

As it has been stated above, by construction, the inner parallelizations of H-DFC and SDP, do not improve the quality solution provided by the related serial versions (if the latter obtains the incumbent solution before reaching the time limit); compare the incumbent solution values in Tables 3 and 4 for each algorithm. However, observe in Table 4 the remarkable reduction in time, IP-SDP obtains greater decreasing but using much higher number of threads. However, the efficiency results are similar for both matheuristics, IP-SDP with 12 threads and IH-DBFC with 2 threads. The efficiency for IP-SDP is between 42% and 79% with an average value of 60%, while the efficiency for IH-DBFC is between 38% and 98% with an average value of 59%. The structure of the scenario tree is not the most appropiate for an inner parallelization of the DBFC algorithm, since it is based on the number of scenario tree nodes $C = |\mathscr{G}^{t^*+1}|$. By construction, the SDP algorithm may solve in parallel as many submodels as there are nodes in *each* stage (and as there are scenarios in the extreme case) . However, notice in the table that the average number of scenario cluster submodels (2) solved at the candidate TNFs, $\overline{c}$, is smaller than the number of submodels C that could be solved at those TNFs (one of the improvements of DBFC over BFC), see subsection 3.1. It is worth to point out that the oracle scheme in Step 4 of DBFC identifies the submodels that are not needed to be solved again because the solutions have already been obtained. For example, in instance c55 (where $t^* = 6$) the number of nodes in set $\mathscr{G}^7$ in the scenario tree is $C = 96$, which means that the solution of all the 96 scenario cluster submodels have to be provided in one way or another. However, the number of submodels that the oracle in Step 4 identifies is, on average, $\overline{c} = 5.43$, whose solution needs to be obtained, since it is not kept from the previous candidate TNF branching. It means that a great deal of time is saved in the serial version. So, a small number of primary threads are needed to obtain the same efficiency as P-SDP in the inner parallel version of H-DBFC.

As a final remark, notice that the chosen cluster distribution criterion within threads aims to balance the number of subproblems. Observe that the instances have symmetric scenario trees, and then, the dimensions of the scenario cluster submodels are quite similar in each instance. So, their solving complexity and elapsed

Table 4: IH-DBFC performance versus IP-SDP

	IP-SDP $(1 \times 12 \times 1)$					IH-DBFC $(1 \times 2 \times 1)$							
Inst	$\overline{z}^{DEM}_{IP-SDP}$	$GG\%$	$t(s)$	S_{12}	$E_{12}\%$	t^*	C	$\overline{c}$	$\overline{z}^{DEM}_{IH-DBFC}$	$GG\%$	$t(s)$	S_2	$E_2\%$
c43	3539594	1.18	5	6.60	55.00	1	3	1.64	3505629	0.21	128	1.26	62.89
c44	4249979	0.92	19	7.63	63.60	1	3	2.09	4226605	0.36	83	1.04	51.81
c45	8123167	1.08	5	5.60	46.67	1	2	1.06	8127924	1.14	196	1.03	51.28
c46	8139108	0.63	8	6.38	53.13	1	2	1.10	8149579	0.76	180	1.03	51.67
c47	7227178	1.06	58	7.47	62.21	2	9	1.92	7175002	0.33	188	1.21	60.37
c48	6660629	1.01	16	6.75	56.25	1	3	2.04	6603906	0.15	185	1.96	98.11
c49	1004692	1.14	3	7.67	63.89	1	2	2.00	993433	$*$	1	1.00	50.00
c50	1006477	0.14	1	6.67	55.56	1	2	1.40	1005119	$*$	4	0.75	37.50
c51	775933	0.44	2	6.67	55.56	1	3	1.60	772589	$*$	14	1.00	50.00
c52	870090	0.85	3	6.00	50.00	1	3	2.00	863562	0.09	12	1.25	62.50
c53	685613	2.29	7	8.43	70.24	1	3	2.00	674479	0.63	276	1.34	67.21
c54	776389	0.93	51	9.53	79.41	1	3	3.00	774450	0.68	158	1.16	57.91
c55	1165132	0.16	9	8.11	67.59	6	96	5.43	1169030	0.49	35	0.80	40.00
c56	1128968	0.24	2	8.33	69.44	1	2	2.00	1129302	0.27	36	1.03	51.39
c57	7256183	1.14	13	7.00	58.33	1	2	1.14	7234093	0.83	180	1.66	83.06
c58	8803020	0.56	26	7.69	64.10	1	2	1.06	8864215	1.26	168	1.26	63.10
c59	8251017	0.61	30	5.70	47.50	1	3	1.50	8253812	0.65	172	1.12	55.81
c79	61360087	–	2850	5.08	42.32	1	3	1.29	61360087	–	11776	1.23	61.61
Average	4097833	0.85	15	7.19	59.91	1	8	1.94	4089572	0.46	119	1.17	58.51

*: Optimality/goodness gap achieved ($< 0.01\%$)
–: non available

time are balanced. Moreover, the number of threads th is very close to the average number $\overline{c}$ of submodels (2) to be required to be solved, then it is more likely to have idle threads rather than unbalanced solving tasks. Notice that given the scheme used in algorithm H-DBFC, the number of submodels to solve per iteration is not constant and then, $\overline{c} < C$ (number of cluster submodels).

5.4. Performance of outer parallelization OH-DBFC

Table 5 shows the main results of the outer parallel OH-DBFC and compares them with OP-SDP, both with the same thread configuration ($12 \times 1 \times 1$). The new headings are as follows: $\overline{z}^{DEM}_{OP-SDP}$ and $\overline{z}^{DEM}_{OH-DBFC}$, incumbent solution values of OP-SDP and OH-DBFC, respectively; $GG\%$ goodness gap, the relative difference (in percentage) between the solution values provided by the outer parallel matheuristics OP-SDP and OH-DBFC, and the CPLEX incumbent solution value, whose expression is $GG\% = 100 \cdot \frac{\overline{z}^{DEM}_{(.)} - z^{DEM}}{z^{DEM}}$, where $(.)$ is OP-SDP and OH-DBFC; $IG\%$, improvement gap of the outer parallel version of algorithm (.) over its related serial version, where (.) is either OH-DBFC or OP-SDP, whose expression is $IG\% = 100 \cdot \frac{\overline{z}^{DEM}_{S-SDP} - \overline{z}^{DEM}_{OP-SDP}}{\overline{z}^{DEM}_{OP-SDP}}$ or $IG\% = 100 \cdot \frac{\overline{z}^{DEM}_{SH-DBFC} - \overline{z}^{DEM}_{OH-DBFC}}{\overline{z}^{DEM}_{OH-DBFC}}$; $\Delta z\%$, the relative gap of OP-SDP over OH-DBFC, that is, the relative difference in percentage between the solution values provided by both algorithms, whose expression is $\Delta z\% = 100 \cdot \frac{\overline{z}^{DEM}_{OP-SDP} - \overline{z}^{DEM}_{OH-DBFC}}{\overline{z}^{DEM}_{OH-DBFC}}$; $\Delta t(s)$, increment (in seconds) of the elapsed time required by OP-SDP over OH-DBFC.

Observe in Table 5 that the quality of the solution in the outer parallelization in either algorithm is improved over their related serial versions in most of the instances. The relative difference on average for

Table 5: OH-DBFC performance versus OP-SDP

	OP-SDP $(12 \times 1 \times 1)$				OH-DBFC $(12 \times 1 \times 1)$					Comparison	
Inst	$\overline{z}^{DEM}_{OP-SDP}$	$GG\%$	$t(s)$	$IG\%$	t^*	$\overline{z}^{DEM}_{OH-DBFC}$	$GG\%$	$t(s)$	$IG\%$	$\Delta z\%$	$\Delta t(s)$
c43	3538641	1.15	**19**	0.03	1	3502309	**0.12**	331	0.09	1.04	-312
c44	4249979	0.92	156	$(-)$	1	4226604	**0.36**	**130**	$(-)$	0.55	26
c45	8099878	**0.79**	**237**	0.29	1	8127924	1.14	249	$(-)$	-0.35	-12
c46	8139979	**0.65**	**59**	-0.01	1	8149579	0.76	270	$(-)$	-0.12	-211
c47	7225961	1.04	**65**	0.02	1	7172544	**0.30**	415	0.03	0.74	-350
c48	6659199	0.99	**82**	0.02	1	6603906	**0.15**	226	$(-)$	0.84	-144
c49	1004692	1.14	23	$(-)$	1	993433	*	**2**	$(-)$	1.13	21
c50	1006476	0.14	4	$(-)$	1	1005149	*	4	$(-)$	0.13	0
c51	775566	0.39	17	**0.05**	1	772589	*	17	$(-)$	0.39	0
c52	869150	0.74	20	0.11	1	863562	**0.09**	**17**	$(-)$	0.65	3
c53	675583	0.80	592	1.48	1	674493	**0.64**	**345**	0.09	0.16	247
c54	775311	0.79	305	0.14	1	773529	**0.5**6	**249**	0.12	0.23	56
c55	1165110	0.16	**97**	$(-)$	3	1163434	**0.01**	724	0.48	0.14	-627
c56	1128801	**0.22**	**31**	0.01	1	1129261	0.27	113	$(-)$	-0.04	-82
c57	7255427	1.13	**115**	0.01	1	7220685	**0.65**	808	0.18	0.48	-693
c58	8801307	**0.54**	**236**	0.02	1	8860548	1.22	249	0.04	-0.67	-13
c59	8246290	0.55	**1328**	0.06	1	8231762	**0.38**	488	0.27	0.18	840
c79	61352973	$-$	**2850**	0.01	1	61135509 [b]	-	17425	-0.03	0.36	-14575
Average	4095138	0.71	**199**	0.13	1	4086548	**0.39**	273	0.08	0.32	-74

$*$: Goodness gap achieved $(< 0.01\%)$
$(-)$: Improvement gap null $(< 0.01\%)$
$-$: non available
[b]: Out of memory (35 Gb)

instances c43 to c59 is 0.13% for OP-SDP and 0.08% for OH-DBFC. The results obtained in instance c79 (whose dimensions are given in Table 2) need to be considered separately. The other instances can be split into two groups, one comprising c43 to c52 (medium sized instances) and the other c53 to c59 (large sized instances). Notice that the solution value obtained by OH-DBFC is better than OP-SDP in 8 out of the 10 instances in group one; and that the former requires less time than the latter in 3 out of the 10 instances while both algorithms require the same time in 2 instances. Observe also that the solution value obtained by OH-DBFC is better than OP-SDP in 5 out of the 7 instances included in group two; and the former requires less time than the latter in 2 out of the 7 instances. It is worth to point out that the larger the instances are, the more time OH-DBFC requires in comparison with OP-SDP, although it gives a better solution value.

Notice that for the largest instance, c79, OH-DBFC provides a better solution value than OP-SDP at the price of requiring much more time. That increase is due to the time required to solve the large LP model (6) that results from fixing at 0-1 values the $x-$variables in the original model (1) at any integer TNF (Step 5 of the algorithm). Notice that no decomposition has been made in the algorithm for satisfying the NAC (4) of the y-variables related to the scenario tree nodes in stages up to break stage t^*, where all 0-1 variables have already been fixed in the related branching. In Section 6, we elaborate on our future plan to address this issue.

6. Conclusions and future work

We have presented the new matheuristic H-DBFC, as a strong improving spin-off from the Branch-and-Fix Coordination (BFC) methodology, and its related inner and outer parallel versions with several matheuristic strategies for solving very large sized multistage stochastic mixed 0-1 problems. A new branching criterion is considered, based on stage-wise ordering and dynamically-guided schemes, such that fewer candidate TNFs are expected to be visited in the algorithm's execution and each of them requires a significant reduction in the time required by its counterpart BFC. It is due to a high number of scenario cluster MIP submodels to be solved in Step 4 for the candidate TNF that are identified by an ad-hoc oracle scheme as already solved in previous iterations of the algorithm and, then, they are not solved again. The outer parallel version performs parallel branching on the 0-1 common variables as well as an iterative exchange of information about the incumbent solution value to reduce the elapsed time. It also obtains tighter bounds on the solution value of the original problem. The use of parallel computing provides a perspective for solving large sized instances on one hand, and a reduction in time on the other hand. The outer parallel H-DBFC can obtain much better results when the number of paths is increased, i.e., when more coordinator threads are considered to work in parallel. A broad computational experience is reported to assess the quality of the matheuristic solution by comparing it with those obtained from plain use of the state-of-the-art CPLEX engine and a different matheuristic algorithm of ours based on Stochastic Dynamic Programming (SDP) [3]. We have experimented with 18 instances from a realistic production planning problem [16]. The instances correspond to symmetric scenario trees and are very well structured. Both matheuristic decomposition algorithms outperform the plain use of CPLEX. The algorithms give solutions with very good quality and H-DBFC provides a better solution value than SDP. However, for reasons stated above, the larger the instances are, the longer the time required by H-DBFC. In any case, the different parallel versions of the matheuristics strongly improve the elapsed time as well as the solution quality compared to the serial versions. As an illustrative example of the computational performance of the new approach, there is a large instance we have experimented with whose dimensions are 448,020 constraints, 116,840 0-1 variables and 272,760 continuous variables. Serial H-DBFC (SH-DBFC) gives a solution value with a 0.49% goodness gap versus plain use of CPLEX v.12.5 and it requires 28 seconds of elapsed time, while CPLEX stops since it was running out of memory (35Gb) after 8,948 seconds whose solution value has a 0.07% quasi-optimality gap at that time instant. The largest instance we have experimented with has the following dimensions: 3,032,055 constraints, 769,500 0-1 variables and 1,919,700 continuous variables. CPLEX stops after reaching the 8h time limit without getting the LP solution. However, SH-DBFC gives a solution value with a 0.39% improvement gap versus Serial SDP (S-SDP) requiring both an elapsed time of 14,511 and 14,473 seconds, respectively.

As future work we are also considering the solution of very large sized problems, by avoiding the potential drawback of solving the compact version of the LP model (6) at any integer TNF (Step 5 of the algorithm). Notice that a matheuristic based on a multistage cluster Lagrangean Decomposition (MCLD) scheme [26] could be considered by dualizing the NAC (4) of the continuous variables related to the scenario tree nodes in the stages up to the break one. The decomposition would consist of solving (serially or in parallel) as many LP submodels as there are scenario clusters generated based on the chosen break stage. The convergence could be accelerated by using a separable quadratic function as a regularization mechanism using the incumbent solution as a reference point. Another direction for future research consists of extending the SH-DBFC algorithm and its parallel versions to consider risk averse strategies as opposed to the risk neutral one considered here. We favour the multifunction multistage time stochastic dominance risk averse strategy introduced in [23] for the reasons presented there. The main challenge consists of handling the cross scenario tree node constraints for selected stages along the time horizon. This could also be done by MCLD of NAC and Lagrangean Relaxation of the risk averse based constraints, see [23]. Another direction for research is the expansion of our approach to allow a mixture of exogenous and endogenous uncertainties in the main parameters.

Acknowledgements

This research has been partially supported by: Bizkaia Talent and European Commission through the COFUND programme, awarded in the 2015 Aid Programme for Researchers with request reference number AYD-000-280; Spanish Ministry of Economy and Competitiveness MINECO and European Regional Development Fund FEDER through the BCAM Severo Ochoa excellence accreditation SEV-2013-0323; projects I+D Excellence MTM2015-65317-P and MTM2015-63710-P; Basque Government through the BERC 2014-2017 program and Grupo de Investigación IT928-16; and University of the Basque Country UPV/EHU through the UFI BETS 2011 programme. The authors thank the technical and human support provided by IZO-SGI SGIker of UPV/EHU and the European funding (ERDF and ESF) as well as the two anonymous reviewers for their help on clarifying some concepts presented in the manuscript and whose positive criticism strongly improved its presentation.

Appendix A. Performance evolution of SH-DBFC and S-SDP

Figure A.1 shows the performance of the matheuristics S-SDP and SH-DBFC on the evolution of the incumbent values in comparison with the value obtained by plain use of CPLEX for the medium sized instances c44 and c47, the larger instances c53, c56 and c57 and the largest instance c79, respectively. The goodness gap is in parenthesis inside the legends.

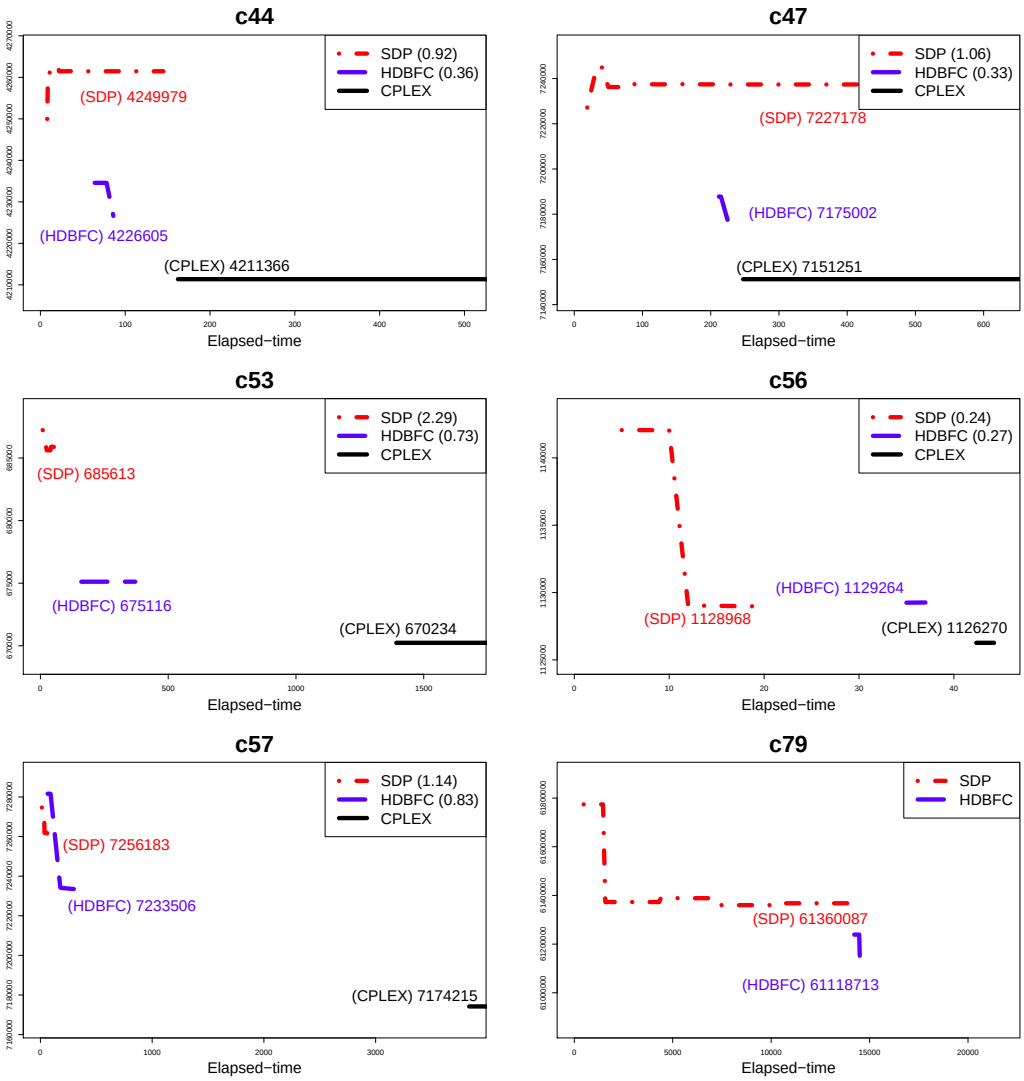

Figure A.1: Comparison of H-DBFC, SDP and CPLEX for instances c44, c47, c53, c56, c57 and c79.

References

[1] T. Al-Khamisl and R. M'Hallah. A two-stage stochastic programming model for the parallel machine scheduling problem with machine capacity. *Computers & Operations Research*, 38:1747–1759, 2011.

[2] U. Aldasoro. *On parallel computing for stochastic optimization models and algorithms*. PhD thesis, University of the Basque Country UPV/EHU, Spain, 2015. URL `https://addi.ehu.es/handle/10810/14315`.

[3] U. Aldasoro, L.F. Escudero, J.F. Monge, M. Merino, and G. Pérez. On Parallelization of a Stochastic Dynamic Programming algorithm for solving large-scale mixed 0-1 problems under uncertainty. *TOP*, 23:703–742, 2015.

[4] U. Aldasoro, L.F. Escudero, M. Merino, and G. Pérez. An algorithmic framework for solving large-scale multistage stochastic mixed 0-1 problems with nonsymmetric scenario trees. Part II: Parallelization. *Computers & Operations Research*, 40:2950–2960, 2013.

[5] L. Aranburu, L.F. Escudero, M.A. Garín, and G. Pérez. Stochastic models for optimizing immunization strategies in fixed-income security portfolios under some sources of uncertainty. In H.I. Gassmann and W.T. Ziemba, editors, *Stochastic Programming: Applications in Finance, Energy, Planning and Logistics*, pages 173–220. World Scientific Publishing Co. Pte. Ltd., 2012.

[6] T. Asamov and W.B. Powell. Regularized Decomposition of High-Dimensional Multistage Stochastic Programs with Markov Uncertainty. *arXiv: 1505,02227*, 2015.

[7] T. Asamov, D. F. Salas, and W. B. Powell. SDDP vs. ADP: The Effect of Dimensionality in Multistage Stochastic Optimization for Grid Level Energy Storage. *arXiv preprint arXiv:1605.01521*, 2016.
[8] J.F. Benders. Partitioning procedures for solving mixed variables programming problems. *Numerische Mathematik*, 14:238–252, 1962.
[9] P. Beraldi, L. Grandinetti, R. Musmanno, and C. Trik. Parallel algorithms to solve two-stage stochastic linear programs with robustness constraints. *Parallel Computing*, 26:1889–1908, 2000.
[10] J.R. Birge. Decomposition and partitioning methods for multistage stochastic linear programs. *Operations Research*, 33:989–1007, 1985.
[11] J.R. Birge and F.V. Louveaux. *Introduction to Stochastic Programming*. 2nd edition, Springer, 2011.
[12] J.R. Birge, C.J. Donohue, D.F. Holmes, and O.G. Svintsitski. A parallel implementation of the nested decomposition algorithm for multistage stochastic linear programs. *Mathematical Programming*, 75:327–352, 1996.
[13] J. Blomval. A multistage stochastic programming algorithm suitable for parallel computing. *Parallel Computing*, 29:431–445, APR 2003.
[14] N. Boland, I. Bakir, B. Dandurand, and A. Erera. Scenario set partition dual bounds for multistage stochastic programming: A hierarchy of bounds and a partition sampling approach. Technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA, 2016.
[15] COIN-OR. COmputational INfrastructure for Operations Research. Website. `http://www.coin-or.org/`.
[16] M.P. Cristóbal, L.F. Escudero, and J.F. Monge. On Stochastic Dynamic Programming for solving large-scale production planning problems under uncertainty. *Computers & Operations Research*, 36:2418–2428, 2009.
[17] D.E. Culler, A. Gupta, and J.P. Singh. *Parallel Computer Architecture: A Hardware/Software Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1997. ISBN 1558603433.
[18] G.B. Dantzig and P.W. Glynn. Parallel processors for planning under uncertainty. *Annals of Operations Research*, 22:1–21, 1990.
[19] A. de Silva and D.A. Abramson. Parallel algorithms for solving stochastic linear programs. In A. Y. Zomaya, editor, *Parallel and Distributed Computing Handbook*. Mc Graw hill, New York, USA, 1996.
[20] B.H. Dias, M.A. Tomin, A.L.M. Mercato, T.P. Ramos, R.B.S. Brandi, I.Ch. da Silva jr., and J.A.P.Filho. Parallel computing applied to stochastic dynamic programming for long term operation planning of hydrothermal power systems. *European Journal of Operational Research*, 229:212–222, 2013.
[21] L.F. Escudero. On a mixture of the Fix-and-Relax Coordination and Lagrangean Substitution schemes for multistage stochastic mixed integer programming. *TOP*, 17:5–29, 2009.
[22] L.F. Escudero, M.A. Garín, M. Merino, and G. Pérez. An algorithmic framework for solving large-scale multistage stochastic mixed 0-1 problems with nonsymmetric scenario trees. *Computers & Operations Research*, 39:1133–1144, 2012.
[23] L.F. Escudero, M.A. Garín, M. Merino, and G. Pérez. On time stochastic dominance introduced by mixed integer-linear recourse in multistage stochastic programs. *European Journal of Operational Research*, 249:164–176, 2016.
[24] L.F. Escudero, M.A. Garín, C. Pizarro, and A. Unzueta. On the strongest lagrangean bounds for stochastic location-assignment problems. *Submitted*, 2016.
[25] L.F. Escudero, M.A. Garín, and A. Unzueta. Cluster lagrangean decomposition for time stochastic dominance induced by mixed integer-linear recourse in multistage stochastic optimization. *Submitted*, 2016.
[26] L.F. Escudero, M.A. Garín, and A. Unzueta. Cluster lagrangean decomposition multistage stochastic optimization. *Computers and Operations Research*, 67:48–62, 2016.
[27] L.F. Escudero, J.L. de la Fuente, C. García, and F.J. Prieto. A parallel computation approach for solving multistage stochastic network problems. *Annals of Operations Research*, 90:1–21, 1999.
[28] L.F. Escudero, M.A. Garín, M. Merino, and G. Pérez. On BFC-MSMIP: an exact branch-and-fix coordination approach for solving multistage stochastic mixed 0-1 problems. *TOP*, 17:96–122, 2009.
[29] L.F. Escudero, M.A. Garín, M. Merino, and G. Pérez. On BFC-MSMIP strategies for scenario cluster partitioning and Twin Node Families branching selection and bounding for multi-stage stochastic mixed integer programming. *Computers & Operations Research*, 37:738–753, 2010.
[30] L.F. Escudero, J.F. Monge, D. Romero Morales, and J. Wang. Expected future value decomposition based bid price generation for large-scale network revenue management. *Transportation Science*, 47:181–197, 2013.
[31] L.F. Escudero, J.F. Monge, and D. Romero Morales. An SDP approach for multiperiod mixed 0–1 linear programming models with stochastic dominance constraints for risk management. *Computers & Operations Research*, 58:32–40, 2015.
[32] L.F. Escudero, J.F. Monge, and D. Romero Morales. On time-consistent stochastic dominance risk averse measure for Tactical Supply Chain Planning under uncertainty. *Submitted*, :, 2016.
[33] D. Gade, G. Hackebeil, S.M. Ryan, J.P. Watson, R.J-B. Wets, and D.L. Woodruff. Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs. *Mathematical Programming*, 157:47–67, 2016.
[34] A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Studies*, 2:82 – 114, 1974.
[35] M. Guignard. Lagrangian relaxation. *TOP*, 11:151–228, 2003.
[36] V. Guigues. SDDP for some interstage dependent risk averse problems and application to hydrothermal planning. *Computational Optimization & Applications*, 57:167–203, 2014.
[37] J.L. Hennessy and D.A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.
[38] IBM. ILOG CPLEX optimizer. Website
. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

[39] P. Kall and S.W. Wallace. *Stochastic Programming*. John Wiley, 1994.
[40] A.J. King and S.W. Wallace. *Modeling with Stochastic Programming.* Springer Series in Operations Research and Financial Engineering, 2012.
[41] E. Klerides and E. Hadjiconstantinou. A decomposition-based stochastic programming approach for the project scheduling problem under time/cost trade-off setting and uncertain durations. *Computers & Operations Research*, 37:2131–2140, 2010.
[42] V. Kozmik and D. P. Morton. Risk-averse stochastic dual dynamic programming. *Optimization Online, org/DB HTML/2013/02/3794*, :, 2013.
[43] H. Li, Z. Lu, and X. Chi. Parallel computing for dynamic asset allocation based on the stochastic programming. In *2010 WASE International Conference on Information Engineering (ICIE)*, volume 2, pages 172–176, Beidaihe, Hebei, China, 2010. IEEE.
[44] X. Li, J. Wei, T. Li, G. Wang, and W.W.-G. Yeh. A parallel dynamic programming algorithm for multi-reservoir system optimization. *Advances in Water Resources*, 67:1 – 15, 2014.
[45] Z. Li and M. Ierapetritou. Capacity expansion planning through augmented lagrangian optimization and scenario decomposition. *AIChE Journal*, 58:871–883, 2012.
[46] J. Linderoth, A. Shapiro, and S. Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142:215–241, 2006.
[47] J.T. Linderoth. Parallel and high performance computing for stochastic programming, 2003. Stochastic Programming. Lecture 13, Atlanta, GA, USA.
[48] M. Lucka, I. Melichercik, and L. Halada. Application of multistage stochastic programs solved in parallel in portfolio management. *Parallel Computing*, 34:469–485, 2008.
[49] G. Lulli and S. Sen. A branch and price algorithm for multistage integer programs with application to stochastic batch sizing problems. *Management Science*, 50:786–796, 2004.
[50] G. Lulli and S. Sen. A heuristic procedure for stochastic integer programs with complete recourse. *European Journal of Operational Research*, 171(3):879–890, 2006.
[51] S. Lumbreras and A. Ramos. Optimal design of the electrical layout of an offshore wind farm applying decomposition strategies. *IEEE Transactions on Power Systems*, 28:1434–1441, 2013.
[52] D. Mahlke. *A Scenario Tree-Based Decomposition for Solving Multistage Stochastic Programs: With Application in Energy Production*. Stochastic Programming. Springer, 2011.
[53] J.M. Mulvey and A. Ruszczyǹski. A new scenario decomposition method for large-scale optimization method. *Operations Research*, 43:477–490, 1995.
[54] S.S. Nielsen and S.A. Zenios. Scalable parallel Benders decomposition for stochastic linear programming. *Parallel Computing*, 23:1069–1088, 1997.
[55] A. Pagès-Bernaus, G. Pérez-Valdés, and A. Tomasgard. A parallelised distributed implementation of a Branch-and-Fix Coordination algorithm. *European Journal of Operational Research*, 244:77–85, 2015.
[56] M.V.F. Pereira and L.M.V.G. Pinto. Stochastic optimization of a multireservoir hydroelectric system, a decomposition approach. *Water Resources Research*, 21:779–792, 1985.
[57] M.V.F. Pereira and L.M.V.G. Pinto. Multistage stochastic optimization applied to energy planning. *Mathematical Programming*, 52:359–375, 1991.
[58] G.Ch. Pflug and A. Pichler. *Multistage Stochastic Optimization*. Springer, 2014.
[59] A. Prekopa. *Stochastic Programming*. Kluwer Academic Publishers, Dordrecht, 1995.
[60] R.T. Rockafellar and R.J-B. Wets. Scenario and policy aggregation in optimisation under uncertainty. *Mathematics of Operations Research*, 16:119–147, 1991.
[61] A. Ruszczyiǹski. Parallel decomposition of multistage stochastic programming problems. *Mathematical Programming*, 58:201–228, 1993.
[62] A. Ruszczyiǹski. On convergence of an augmented lagrangian decomposition method for sparse convex optimization. *Mathematics of Operations Research*, 20:634–566, 1995.
[63] A. Ruszczyiǹski and A. Swietanowski. Accelerating the regularized decomposition method for two stage stochastic linear problems. *SIAM Journal on Optimization*, 24:127–153, 2014.
[64] B. Sandikci and O.Y. Ozaltin. A scalable bounding method for multi-stage stochastic integer programs. Technical report, Working paper 14-21, Booth School of Business, University of Chicago, Chicago, IL, USA, 2014. URL `http://dx.doli.org/10.21.39/ssrn.26666`.
[65] B. Sandikci, N.Kong, and A.J. Schaefer. A hierarchy of bounds for stochastic mixed-integer programs. *Mathematical Programming*, 138:253–272, 2013.
[66] S. Sen and Z. Zhou. Multistage stochastic decomposition: a bridge between stochastic programming and approximate dynamic programming. *SIAM Journal on Optimization*, 24:127–153, 2014.
[67] A. Shapiro, D. Dentcheva, and A. Ruszczyǹski. *Lectures on stochastic programming: Modeling and theory.* MPS-SIAM Book Series on Optimization 9, 2009.
[68] A. Shapiro, W. Tekaya, J.P. da Costa, and M. Pereira. Risk neutral and risk averse stochastic dual dynamic programming method. *European Journal of Operational Research*, 224:375–391, 2013.
[69] R. van Slike and R.J.B. Wets. L-shaped linear programs with application to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17:638–663, 1969.
[70] H. Vladimirou and S.A. Zenios. Scalable parallel computations for large-scale stochastic programming. *Annals of Operations Research*, 90:87–129, 1999.

[71] S.W. Wallace and W.T. Ziemba (eds.). *Applications of Stochastic Programming.* MPS-SIAM Book Series on Optimization 5, 2005.
[72] J.P. Watson and D. Woodruff. Progressive hedging innovations for a class of stochastic mixed-integer resource allocation problems. *Computational Management Science*, 8:355–370, 2011.
[73] R.J-B. Wets. Stochastic programs with fixed recourse: The equivalent deterministic program. *SIAM Review*, 16:309–339, 1974.
[74] G.L. Zenarosa, O.A. Prokopyev, and A.J. Schaefer. Scenario-tree decomposition: Bounds for multistage stochastic mixed-integer programs. Technical report, Technical paper, Department of Industrial Engineering, University of Pittsburgh, Pittsburgh, PA, USA, 2014. URL `http://dx.doli.org/10.21.39/ssrn.26666`.
[75] J. Zou, S. Ahmed, and X.A. Sun. Nested Decomposition of Multistage Stochastic Integer Programs with Binary State Variables. Technical report, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, USA, 2016.