Noname manuscript No.
(will be inserted by the editor)

A Benders squared (B^2) framework for infinite-horizon stochastic linear programs

Giacomo Nannicini · Emiliano Traversi · Roberto Wolfler Calvo

Received: date / Accepted: date

Abstract We propose a nested decomposition scheme for infinite-horizon stochastic linear programs. Our approach can be seen as a provably convergent extension of stochastic dual dynamic programming to the infinite-horizon setting: we explore a sequence of finite-horizon problems of increasing length until we can guarantee convergence with a given confidence level. The methodology alternates between a forward pass to explore sample paths and determine trial solutions, and a backward pass to generate a polyhedral approximation of the optimal value function by computing subgradients from the dual of the scenario subproblems. A computational study on a large set of randomly generated instances for two classes of problems shows that the proposed algorithm is able to effectively solve instances of moderate size to high precision, provided that the instance structure allows the construction of what we call constant-state policies with satisfactory objective function value.

Keywords Benders Decomposition · Stochastic Programming · Cutting Planes

1 Introduction

In the context of optimal planning under uncertainty, there are many situations in which the decision maker is interested in solving a steady-state planning problem.

G. Nannicini
IBM T.J. Watson, Yorktown Heights, NY
E-mail: nannicini@us.ibm.com

E. Traversi
Laboratoire d'Informatique de Paris Nord,
Université de Paris 13; and Sorbonne Paris Cité,
CNRS (UMR 7538), 93430 Villetaneuse, France
E-mail: traversi@lipn.fr

R. Wolfler Calvo
Laboratoire d'Informatique de Paris Nord,
Université de Paris 13; and Sorbonne Paris Cité,
CNRS (UMR 7538), 93430 Villetaneuse, France
E-mail: wolfler@lipn.fr

Such a scenario arises whenever there are repeated decisions that have to be taken over an infinite (i.e. unbounded) time horizon, for example if production of a given set of items has to be planned every day under uncertain demands, and this process is repeated indefinitely. In this type of model there is a discrete time index t that ranges from zero (present time) to infinity, and a decision problem must be solved for every integer value of t (called a *stage*). The decision problem at each stage depends on the *state* of the system, which is influenced by both the realization of uncertain events, and previous decisions (often called *actions*). For example, the state of the system could be the inventory of the items at hand, and the actions could be the set of items ordered from a supplier at that point in time. Each action incurs a cost, and the objective is to find a rule that maps states into actions (called a *policy*) in order to minimize the total expected cost over the time horizon $t = 0, \ldots, \infty$. A geometric discount factor for the cost is typically introduced, to model the fact that the present-time value of decisions that will be taken in the far future is small, and goes to zero the longer the distance in time; this guarantees finiteness of the cost under mild conditions. A problem of this form is naturally described as a dynamic program (DP, to be distinguished from "dynamic programming" by context), introduced in the seminal work [2]. Well-known methodologies to solve infinite-horizon DPs are value iteration and policy iteration, which converge to an optimal solution under some assumptions, see [3]. These methodologies however encounter considerable difficulties when the state space (the set of possible states) is infinite, because in that case the basic iteration step of these algorithms is no longer able to loop over all the states [26, Ch. 3].

A class of DPs exhibiting infinite state and action spaces is that of multistage stochastic linear programs, see e.g. [5,6,30]. This type of problems finds many applications within the broad context of optimal planning under uncertainty. Examples of such applications are financial planning [22,23,31] and capacity expansion [6, S. 1.3]. Despite their infinite state and action spaces, multistage stochastic LPs can be solved using various methodologies under the assumption that the time horizon is finite: a widely used approach is Stochastic Dual Dynamic Programming (SDDP), first proposed in [24] and with many follow-up works, e.g. [9]. SDDP is essentially a Benders decomposition scheme that assumes that the sample space Ω is discrete and finite, and the realizations of uncertainty are time-wise (interstage) independent. In case the sample space is not discrete and finite, sampling can be used to reduce to the finite case, as discussed in [29], where error bounds are provided.

This paper presents a Benders decomposition scheme for infinite-horizon stochastic LPs that is provably convergent to an optimal policy; we call this algorithm B^2 (Benders squared). B^2 can be seen as an extension of SDDP to the infinite-horizon setting, and it works under similar assumptions: the sample space Ω is discrete and finite, and the realizations of uncertainty are not only interstage independent but also identically distributed (i.i.d.). Notice that some degree of (linear) stagewise dependence can be introduced by modifying the problem formulation extending the state variables just as in SDDP, see e.g. [29]. Requiring identically distributed data at each stage is natural for an infinite-horizon problem when one is interested in a steady-state solution, because such solution may not be reached if the data process changes over time. B^2 follows a typical Benders scheme for multistage

problems, but it dynamically determines the length of the time horizon to explore, and exploits the i.i.d. assumption to strengthen the cut generation problems.

We test B^2 on two classes of randomly generated (continuous) problems: a classical multi-product production planning problem, and the rebalancing problem. The rebalancing problem arises e.g. in the context of bike sharing systems, where we are looking for a policy to optimally rebalance the availability of bikes at a given set of stations, subject to uncertain demand. Notice that these problems are often stated with integrality restrictions on the decision variables, whereas in this paper we aim to solve their continuous relaxation. However, the solution of multi-stage and infinite-horizon problems with discrete variables is notoriously difficult, and often practically impossible, due to nonconvexities. For this reason, a commonly employed heuristic strategy consists in determinining a polyhedral description of the optimal policy for the continuous relaxation of the problem, adding integrality restrictions on the decision variables only at a later time to determine a feasible (and most likely suboptimal) policy. If the original problem involves discrete variables, the approach presented in this paper can be used as a heuristic in this manner. Numerical experiments show that our algorithm is considerably more effective than applying SDDP to a truncated problem, and is able to solve medium-sized instances (tens of variables, constraints, and scenarios) to high accuracy, i.e. with small gaps between lower and upper bounds. This suggests that the methodology could scale to larger instances (hundreds of variables and constraints) provided the user is willing to accept lower accuracy. Since our methodology makes extensive use of Benders cuts, our computational evaluation sheds some light on which implementation details are more effective in practice, in particular showing that a multicut implementation [7] converges much faster than a single cut implementation, which is predominant in the literature, e.g. [24, 19, 33].

The rest of this paper is organized as follows. Section 2 formally describes the problem studied in this paper and states our assumptions. Section 3 paves the way for a decomposition algorithm. Section 4 presents a solution methodology to obtain dual bounds, discusses the computation of primal bounds, and the convergence properties of our scheme. Section 5 provides a numerical evaluation of the proposed methodology, and Section 6 concludes the paper. The rest of this section overviews some of the existing literature that is most relevant for our paper.

Literature review. The area of multistage stochastic LP has received considerable attention in the mathematical optimization literature, due to its many real-world applications. [14] studies how an infinite-horizon LP can be approximated by a finite horizon problem, but the paper considers a deterministic setting rather than a stochastic setting. In [27], an extension of LP duality theory to infinite-dimensional problems is studied. However, the literature discussing practical solution strategies typically focuses on finite problems, i.e., finite horizon, finite number of variables and constraints.

The most commonly used solution methodologies exploit LP duality to devise some form of decomposition scheme, see [5, 24, 30]. Typically, the data of the problem is assumed to be interstage independent, which facilitates the decomposition scheme: this is also the assumption in the present paper. However, some work that addresses interstage dependency can be found in the literature: for example, [17]

studies how to adapt Benders cuts to the case of a model with interstage dependence, while [15] proposes a SDDP scheme that can take into account some types of interstage dependency. The popularity of SDDP (and Benders decomposition in general) prompted computational studies on how to generalize and improve its performance: particularly relevant for our work are the papers [8], which discusses the tuning of SDDP to solve a specific instance and some general-purpose cut selection strategies, and [7], which investigates the issue of multicut versus single cut on a small set of test problems. Some papers proposed to augment the Benders subproblem with a regularization term: for example, [1] studies the impact of a quadratic regularization terms when solving high-dimensional multistage stochastic programs, and [13] shows how to generalize SDDP to solve nonlinear convex problems. There have also been studies about sequencing protocols to choose the order in which subproblems of the decomposition scheme are solved, see [5,9,11]. To deal with problems with a scenario tree that is too large to handle, a methodology to select an ϵ-optimal subtree of the scenario tree is given in [16]. While nested decomposition approaches are usually formulated for continuous problem, they can be extended to integer problems under some conditions, as in the recent work [33]. It is likely that some of the techniques discussed in these papers can also be applied to our setting, because the underlying algorithm closely mimicks SDDP.

2 Problem formulation and assumptions

We study a sequential stochastic decision-making problem in which time evolves in a discrete manner over an infinite horizon. Information carried from one stage to the next is encoded by a state. At each stage t we need to determine x_t, representing the action taken, and y_t, representing the state in which we leave the system. The set of feasible states and actions is encoded by linear constraints. We assume that actions and state incur a linear cost of the form $c^\top x_t + h^\top y_t$ at each stage t, with a discount factor $\delta < 1$ for future costs. The vector of right-hand side values R_t is random, and it takes values from the sample space Ω. Our goal is to compute a policy that maps states to actions and achieves minimum total cost; a more precise formulation is given later in this section, see (IHLP). A possible interpretation of such a model is that of optimally planning production of a certain set of items over an infinite horizon, with uncertain demand R_t. At each stage we need to determine a vector x_t of production decisions, as well as a carryover vector y_t that represents inventory or backlogged demand.

We make the following assumptions, discussed below:

(A1) $\Omega = \{\omega_1, \dots, \omega_\kappa\}$ with $\kappa < \infty$.
(A2) There exists $U < \infty$ and an integer θ such that imposing the constraints $\|x_t\|_\infty \leq Ut^\theta, \|y_t\|_\infty \leq Ut^\theta$ does not change the solution to the problem.
(A3) $R_t, t = 0, \dots, \infty$, are i.i.d.
(A4) The problem has relatively complete recourse, i.e., at every stage $t = 1, \dots, \infty$, for every feasible y_{t-1}, there exists a feasible choice of x_t, y_t.
(A5) There is no feasible direction with unbounded cost for $t = 0, \dots, \infty$ and $\omega \in \Omega$.

Assumption (A1) enforces finiteness of the sample space possibly by taking a finite number of samples from the original space, see e.g. [29]. It is a standard as-

sumption for approaches based on Benders decomposition [6]. Assumption (A2), ensuring that x_t and y_t grow at most polynomially with t, is a sufficient condition to show that the total cost is bounded. Assumption (A3) is fundamental for the computational efficiency of the B^2 algorithm, which is based on SDDP. As mentioned in the introduction, this assumption ensures that the data process has a steady state, and some degree of (linear) stagewise dependence can be introduced extending the state variables [29]. Assumption (A4) simplifies our exposition and avoid dealing with hard-to-detect infeasibilities of the mathematical optimization problem. Assumption (A5) ensures that the costs $\delta^t(c^\top x_t + h^\top y_t)$ are uniformly bounded, see e.g. [27].

Let $K := \{1, \dots, \kappa\}$. The realizations of uncertainty $\{\omega_1, \dots, \omega_\kappa\}$ are also called "scenarios" in the following. Due to nonanticipativity, the number of possible states and therefore of possible decisions that can be taken at time t is equal to the number of sample paths up to time t, which is $|K|^t$ under our assumptions. Following the general description given at the beginning of this section, the vector of right-hand side values at time t is a random variable $R_t(\omega) = (b_t(\omega), d_t(\omega), w_t(\omega))$. For $k \in K$, denote $p_k = \Pr(\omega_k), b_k = b(\omega_k), d_k = d(\omega_k), w_k = w(\omega_k)$, since the time index is not necessary for b, d, w due to independence. For $t = 0, \dots, \infty$, let $S_t := K^t = \underbrace{K \times K \times \cdots \times K}_{t \text{ times}}$; hence, S_t represents the possible sample paths up to stage t. For $s \in S_\tau$, we denote by p_s the probability of occurrence of all sample paths that concide with s up to stage τ; by (A3), $p_s = \prod_{t=1,\dots,\tau} p_{s_t}$. Given $s \in S_\tau$, we denote by $\underline{s} := (s_1, \dots, s_{\tau-1})$, i.e. the τ-tuple s truncated to the $(\tau-1)$-th element. We index $s \in S_\tau$ as a vector, e.g. s_τ is the last element of s. Throughout this paper, for notational convenience we assume that $s_0 = 0$ and the special scenario index "0" is used to refer to the initial (deterministic) conditions of the problem. We can now precisely state the problem addressed in this paper:

$$
\begin{aligned}
\text{(IHLP)} \qquad \min \quad & c^\top x_0 + h^\top y_0 + \sum_{t=1}^{\infty} \delta^t \sum_{s \in S_t} p_s (c^\top x_t(s) + h^\top y_t(s)) \\
t = 0, \dots, \infty, s \in S_t \quad & A x_t(s) - T y_{t-1}(\underline{s}) + G y_t(s) \geq b_{s_t} \\
t = 0, \dots, \infty, s \in S_t \quad & D x_t(s) \geq d_{s_t} \\
t = 0, \dots, \infty, s \in S_t \quad & W y_t(s) \geq w_{s_t}
\end{aligned}
$$

where $A \in \mathbb{R}^{m \times n_1}$, $G \in \mathbb{R}^{m \times n_2}$, $T \in \mathbb{R}^{m \times n_2}$, $D \in \mathbb{R}^{p \times n_1}$, $W \in \mathbb{R}^{q \times n_2}$, $0 \leq \delta < 1$, $b_k \in \mathbb{R}^m$, $d_k \in \mathbb{R}^p$, $w_k \in \mathbb{R}^q$, and the matrices D, W subsume lower and upper bounds on the decision variables. In the formulation (IHLP), which stands for "Infinite-Horizon Linear Program", we allow a different decision $x_t(s)$ and state $y_t(s)$ for every sample path. Note that this automatically ensures nonanticipativity. The formulation (IHLP) is an infinite-horizon discounted linear program. This problem is called "doubly-infinite" because it has an infinite number of decision variables and constraints [27], but just as in [27], each variable appears in a finite number of constraints, and each constraint contains a finite number of variables, see also [28]. In (IHLP), uncertainty only affects the right-hand side vectors of the mathematical program. It may be possible to generalize B^2 to the case in which the matrices A, G, T, D, W are also affected by uncertainty, but this is not

straightforward because, at a minimum, it entails keeping track of which of the generated inequalities (discussed later in the paper) are valid for each scenario.

The constraints of (IHLP) describe the evolution of the system through time. In particular, the variables y_t represent the state of the system from the dynamic programming point of view, see e.g. [3], because for any given τ, problem (IHLP) is completely determined for time $t = \tau, \ldots, \infty$ given the realization R_τ and $y_{\tau-1}$. Notice that in our notation, at stage τ the endogenous part of the state is given by $y_{\tau-1}$ rather than y_τ, because y_τ is a decision that is taken at time τ. Here and throughout the paper, the vector y_{-1} represents the initial state of the system and is part of the problem input.

It is well known that under our assumptions the problem admits an optimal stationary policy, see e.g. [3]. More specifically, regardless of the initial conditions R_0, there exists an optimal *value function* $V^{*,k}(\bar{y})$ for each scenario k that gives the optimal *cost-to-go*, i.e. the minimum discounted cost incurred until the end of the time horizon starting in state $\bar{y}$ with uncertainty realization ω_j. The optimal policy for realization ω_k is then simply given by the solution of the Bellman equations [2], which in this specific case read:

$$\begin{aligned} V^{*,k}(\bar{y}) := \min \quad & c^\top x + h^\top y + \delta \sum_{j \in K} V^{*,j}(y) \\ & Ax - T\bar{y} + Gy \geq b_k \\ & Dx \geq d_k \\ & Wy \geq w_k \end{aligned} \tag{1}$$

One can also define the average value function $V^*(\bar{y}) = \sum_{k \in K} p_k V^{*,k}(\bar{y})$, giving the expected cost of being in state $\bar{y}$. Conversely, any stationary policy that attains the above minimum for every state $\bar{y}$ is optimal.

3 Benders decomposition: preliminaries

Problem (IHLP) has infinite size, and therefore it cannot be solved directly by means of linear programming tools. However, under assumptions (A1)-(A5) the optimal cost of (IHLP) is bounded: the present value of future costs decreases following a geometric series with a discount factor $\delta < 1$, whereas the optimal solution value at each stage increases at most polynomially. Hence, we are interested in a methodology to determine the optimal cost and the optimal policy.

In this paper we propose a solution methodology based on Benders decomposition. Benders decomposition has been successfully applied to a finite-horizon version of (IHLP), for example in SDDP [24] and in the L-shaped method [30]. We refer to [6] for a more detailed discussion of decomposition methods in stochastic programming, including nested decomposition methods for multi-stage problems. Note that a possible way to solve (IHLP) within a certain optimality tolerance $\epsilon > 0$ is to determine a large enough truncation τ of the time horizon so that the objective function contribution becomes sufficiently small, i.e. $\sum_{t=\tau}^{\infty} \delta^t \sum_{s \in S_t} p_s (c^\top x_t(s) + h^\top y_t(s)) \leq \epsilon$. Then, solving the truncated problem, for example via SDDP, yields a solution with error at most ϵ. The desired τ exists due to the boundedness assumption, but it may not be easy to determine its value.

Moreover, in practice it is likely that τ will be very large if ϵ is small, therefore it would be very inefficient or even practically impossible to compute the solution to the truncated problem using a methodology for finite-horizon problems, and a tailored decomposition scheme is desirable. The algorithm that we propose is based on SDDP, and can be seen as its extension to the infinite-horizon case.

In a Benders decomposition scheme, the decision variables are assigned to a master problem and/or to one or several slave problems. Each subproblem only keeps the constraints involving at least one of its variables. Assuming that all variables that contribute to the objective function are in the master (as is the case in this paper, after a reformulation), a solution of the master problem is optimal if it can be completed to a feasible solution for all slave problems. To ensure feasibility of the slave problems we rely on properties of linear programming duality and iteratively derive inequalities that can be added to the master, see e.g. [4]; in particular, the dual of the slave problems must be bounded. We remark that, because our problem is doubly-infinite, strong duality may not hold in general: see e.g. [10], which points out the existence of a duality gap in case some interior point conditions fail to hold. However, an advantage of the approach presented in this paper is that the algorithm works with a sequence of finite-dimensional linear programs only, and the convergence proof relies on well-known properties of dynamic programs and a classical cutting plane approach [18]. For this reason, we do not have to rely on sufficient conditions for strong duality [27].

It will be convenient to introduce auxiliary variables to represent upper bounds on the one-stage costs; this way, all slave problems are feasibility problems. From now on, $z_t(s)$ denotes an upper bound on the cost from time t to ∞ starting from sample path $s \in S_t$. We obtain the following formulation, which is easily seen to be equivalent to (IHLP):

$$
\begin{aligned}
&\text{(IHLP-F)} \qquad \min && c^\top x_0 + h^\top y_0 + \delta \sum_{k \in K} p_k z_1(k) \\
&t = 0, \dots, \infty, s \in S_t && Ax_t(s) - Ty_{t-1}(\underline{s}) + Gy_t(s) \geq b_{s_t} \\
&t = 0, \dots, \infty, s \in S_t && Dx_t(s) \geq d_{s_t} \\
&t = 0, \dots, \infty, s \in S_t && Wy_t(s) \geq w_{s_t} \\
&t = 1, \dots, \infty, s \in S_t && z_t(s) - c^\top x_t(s) - h^\top y_t(s) - \delta \sum_{k \in K} p_k z_{t+1}(s \times \{k\}) \geq 0
\end{aligned}
$$

We label this problem (IHLP-F) for "Infinite-Horizon Linear Program – Feasibility version". We can apply Benders decomposition on this problem, solving a master problem with variables x_0, y_0, z_1, keeping the constraints for $t = 0$ only and ensuring feasibility for the remaining constraints through a set of Benders cuts C obtained from the dual variables of the slave problem. Note that the slave problem has an infinite number of variables and constraints, therefore its feasible region may not be representable by a finite number of Benders cuts, see Sect. 4. However, as noted earlier in this section, there exists a truncation of the time horizon that yields an ϵ-optimal solution, and we will implicitly work with such a truncation. Hence, for now we simply assume that we aim at ensuring feasibility of the slave problem with a finite number of Benders cuts: Sect. 4.3 shows that our methodology indeed converges to an ϵ-optimal solution. The proposed decomposition scheme yields the following master problem (denoted (MP-L0) for "Master

Problem, Level 0"):

$$\begin{aligned}
\text{(MP-L0)} \qquad \min \quad & c^\top x_0 + h^\top y_0 + \delta \sum_{k \in K} p_k z_1(k) \\
& Ax_0 + Gy_0 \geq b_0 + Ty_{-1} \\
& Dx_0 \geq d_0 \\
& Wy_0 \geq w_0 \\
\forall \gamma \in C \quad & \gamma_y^\top y_0 + \sum_{k \in K} \gamma_k z_1(k) \geq \gamma_0.
\end{aligned}$$

Since all constraints in (IHLP-F) that contain x_0 are kept in (MP-L0), the Benders cuts in C do not involve the variables x_0, but they can involve the other variables. Furthermore, if we look at the constraints in (IHLP-F) that are not part of (MP-L0), we notice that at any given time step τ we can decompose the remaining constraints into κ independent problems, since decision variables for scenario $k \in K$ do not interact with decision variables from any other scenario $k' \neq k$. Hence, we can decompose the slave problem into κ feasibility subproblems, which we state for a generic time index τ (assume $\tau = 1$ for now) and scenario index $j \in K$:

$$\begin{aligned}
\text{(SP-L}\tau\text{-}j\text{)} \qquad \min \quad & 0 \\
& Ax_\tau + Gy_\tau \geq b_j + T\hat{y}_{\tau-1} \\
& Dx_\tau \geq d_j \\
& Wy_\tau \geq w_j \\
& -c^\top x_\tau - h^\top y_\tau - \delta \sum_{k \in K} p_k z_{\tau+1}(j \times \{k\}) \geq -\hat{z}_\tau(j) \\
t = \tau+1, \ldots, \infty, s \in S_t \quad & Ax_t(s) - Ty_{t-1}(\underline{s}) + Gy_t(s) \geq b_{s_t} \\
t = \tau+1, \ldots, \infty, s \in S_t \quad & Dx_t(s) \geq d_{s_t} \\
t = \tau+1, \ldots, \infty, s \in S_t \quad & Wy_t(s) \geq w_{s_t} \\
t = \tau+1, \ldots, \infty, s \in S_t \quad & z_t(s) - c^\top x_t(s) - h^\top y_t(s) - \delta \sum_{k \in K} p_k z_{t+1}(s \times \{k\}) \geq 0.
\end{aligned}$$

This problem is denoted (SP-Lτ-j) for "Slave Problem, Level τ, scenario j". Notice that according to (IHLP-F), x_τ, y_τ should in principle depend on the sample path up to stage τ, but mimicking the notation of (MP-L0) we do not make such dependency explicit because the sample path up to τ is already determined when subproblem (SP-Lτ-j) is considered. In the case of the slave problems for (MP-L0), $\tau = 1$, $j \in K$ is fixed and $\hat{y}_0, \hat{z}_1(j)$ are given from the master (MP-L0). Because of assumption (A4), the only possible infeasibility in (SP-Lτ-j) with $\tau = 1$ is a violation of the upper bound on the objective function imputed from the master $\hat{z}_1(j)$. Problem (SP-Lτ-j) for $\tau = 1$ is again a doubly-infinite linear program, and we want to solve it with the same decomposition scheme: we consider a master problem with the variables $x_1, y_1, z_2(j)$ only, keep the constraints for $t = 1$ and delegate the rest to the slave problems, one for each scenario. Feasibility for the slave problems is ensured via set of Benders cuts C, which are added to the master.

Iterating this idea for a generic time index τ and solving (SP-Lτ-j) by Benders decomposition, the master problem of the level τ slave for a given scenario $j \in K$

with a set of cuts C is the following problem, where we indicate labels for the dual variables corresponding to each set of constraints within square brackets on the right:

$$
\begin{aligned}
\text{(MP-L}\tau\text{-}j\text{)} \quad \min \quad & 0 & \\
& Ax_\tau + Gy_\tau \geq b_j + T\hat{y}_{\tau-1} & [\mu_y] \\
& Dx_\tau \geq d_j & [\mu_d] \\
& Wy_\tau \geq w_j & [\mu_w] \\
& -c^\top x_\tau - h^\top y_\tau - \delta \sum_{k\in K} p_k z_{\tau+1}(j \times \{k\}) \geq -\hat{z}_\tau(j) & [\mu_z] \\
\forall \gamma \in C \qquad & \gamma_y^\top y_\tau + \sum_{k\in K} \gamma_k z_{\tau+1}(j \times \{k\}) \geq \gamma_0. & [\mu_\gamma]
\end{aligned}
$$

Notice that in principle we have an infinite number of these problems, each of which should have its own set of Benders cuts that come from the respective slave problems. Furthermore, so far we have not shown that $|C|$ is finite, but clearly this is the case if the time horizon is truncated at some point, see Sect. 4.3.

The decomposition algorithm that we propose always generates Benders cuts from problems of the form (MP-Lτ-j). The following proposition establishes validity of the Benders cuts generated from any given (MP-Lτ-j) with fixed $j = \hat{\jmath}$ for every other problem (MP-Lτ-j). Since (MP-Lτ-j) is a relaxation of (SP-Lτ-j), the Benders cuts are valid for (SP-Lτ-j).

Proposition 1 *Let $\hat{\mu}$ be an unbounded ray of the dual of* (MP-Lτ-j) *with $\tau = \hat{\tau}$, $j = \hat{\jmath}$, $C = \hat{C}$. Then the inequality $\hat{\mu}_y^\top T y_\tau - \hat{\mu}_z z_{\tau+1}(j) \leq -\hat{\mu}^\top b_{\hat{\jmath}} - \hat{\mu}_d^\top d_{\hat{\jmath}} - \hat{\mu}_w^\top w_{\hat{\jmath}} - \sum_{\gamma\in\hat{C}} \hat{\mu}_\gamma^\top \gamma_0$ is valid for problem* (MP-Lτ-j) *for every τ, j and $C \supseteq \hat{C}$, including* (MP-L0).

Proof The dual of (MP-Lτ-j) is:

$$
\begin{aligned}
\text{(D-MP-L}\tau\text{-}j\text{)} \quad \max \quad & \mu_y^\top (b_j + T\hat{y}_{\tau-1}) + \mu_d^\top d_j + \mu_w^\top w_j - \mu_z \hat{z}_\tau(j) + \sum_{\gamma\in C} \mu_\gamma^\top \gamma_0 \\
& \mu_y^\top A + \mu_d^\top D - \mu_z^\top c = 0 \\
& \mu_y^\top G + \mu_w^\top W - \mu_z^\top h + \sum_{\gamma\in C} \mu_\gamma \gamma_y = 0 \\
k \in K \qquad & -\mu_z \delta p_k + \sum_{\gamma\in C} \mu_\gamma \gamma_k = 0 \\
& \mu_y, \mu_d, \mu_w, \mu_z, \mu_\gamma \geq 0
\end{aligned}
$$

We first show validity for $\tau = \hat{\tau}$, $j = \hat{\jmath}$, $C = \hat{C}$, and then argue why we can generalize to every τ, j and $C \supseteq \hat{C}$, as well as (MP-L0).

Feasibility for problem (MP-Lτ-j) requires that its dual is feasible and bounded. Clearly (D-MP-Lτ-j) is feasible as it admits the all-zero solution. Hence, for every unbounded ray $\hat{\mu}$ of (D-MP-Lτ-j) we must impose that the ray makes a negative angle with the objective function, i.e. $\hat{\mu}_y^\top (b_j + T\hat{y}_{\tau-1}) + \hat{\mu}_d^\top d_j + \hat{\mu}_w^\top w_j - \hat{\mu}_z \hat{z}_\tau(j) + \sum_{\gamma\in C} \hat{\mu}_\gamma^\top \gamma_0 \leq 0$. Remembering that the ray $\hat{\mu}$ comes from (MP-Lτ-j)

with $j = \hat{\jmath}, C = \hat{C}$ and that the only decision variables coming from the previous-level problem in the expression above are $\hat{y}_{\tau-1}$ and $\hat{z}_\tau(j)$, we obtain the desired inequality with the time index shifted by 1.

We now show that the above inequality is valid for (MP-Lτ-j) every τ, j and $C \supseteq \hat{C}$. First, notice that the description of the feasible region of this problem depends neither on τ nor on $j \in K$, and is therefore the same for all problems (MP-Lτ-j) provided C does not change. Then, it is easy to verify that solutions to (D-MP-Lτ-j) are feasible for the dual of (MP-Lτ-j) with any larger set of cuts $C \supseteq \hat{C}$, as the dual variables for the new cuts can simply be fixed to zero. This implies that the inequality derived from an unbounded ray is valid regardless of the choice of τ, j and $C \supseteq \hat{C}$.

To conclude the proof, we note that (MP-Lτ-j) for $\tau = 1$ is a slave problem for the Benders decomposition in which (MP-L0) is the master. Thus, the inequality is valid for (MP-L0) as well. □

From a computational perspective, the important consequence of Prop. 1 is that a Benders cut generated at any level of the decomposition scheme, i.e. for any τ, can be added to all levels of the decomposition. This is in contrast to the classical SDDP approach, where cuts are generated in a backward fashion starting from the last time period, and added only to the master problems of the immediately preceding time period. The possibility of sharing the same pool of Benders cuts among the different levels of the decomposition is due to the fact that we are working on an infinite-horizon, in which case the shape of the value function is the same, regardless of the specific time period. This is not the case in the classical SDDP approach, where the time horizon is finite. On the other hand, we remark that if the matrices A, D, G, W depend on j rather than being fixed, then a cut generated according to Prop. 1 is valid only for problems (MP-Lτ-j) with the corresponding $\hat{\jmath}$.

4 Description of the algorithm and convergence

We now describe in detail our decomposition algorithm, called B^2, discuss the computation of primal bounds, and prove convergence of B^2.

4.1 Description of B^2

B^2 relies on a sampled nested Benders decomposition scheme. The algorithm works in a similar way to SDDP, alternating between a *forward pass* that determines a sequence of trial solutions following a sample path, and a *backward pass* that improves the value function approximation at the trial solutions. A pseudocode description of algorithm B^2 is given in Algorithm 1. The auxiliary problems solved by the algorithm are defined below: $P(\bar{b}, \bar{d}, \bar{w}, C)$ is the problem solved in the forward pass, $RB(\bar{b}, \bar{d}, \bar{w}, \bar{z}, C)$ the problem solved in the backward pass. The label "P" stands for "Primal", "RB" for "Relaxed Benders". Note that $RB(\bar{b}, \bar{d}, \bar{w}, \bar{z}, C)$

Algorithm 1 Infinite-horizon B^2 algorithm.

$C \leftarrow \emptyset$, $\tau \leftarrow 0$, $LB \leftarrow -\infty, UB \leftarrow +\infty, S^* \leftarrow \emptyset$, $n \leftarrow 0$
while stopping criterion not satisfied **do**
  $\tau \leftarrow \tau + 1$ /* This is the length of the time horizon under consideration */
  Choose $\tilde{S}_\tau \subseteq S_\tau$.
  for $s \in \tilde{S}_\tau$ **do**
    $n \leftarrow n + 1$ /* This is the iteration number, i.e., number of passes */
    for $t = 0, \ldots, \tau$ **do**
      Solve $P(b_{s_t} + T\hat{y}_{t-1}(s), d_{s_t}, w_{s_t}, C)$. Let $(\hat{x}_t(s), \hat{y}_t(s), \hat{z}_{t+1}(s))$ be the solution.
    for $t = \tau, \ldots, 1$ **do**
      Generate Benders cuts from $RB(b_{s_t} + T\hat{y}_{t-1}(s), d_{s_t}, w_{s_t}, \hat{z}_{t+1}(s), C)$.
      Add cuts to C.
  for $s \in S^*$ **do**
    Sample $k \in K$ and extend $s \leftarrow s \times \{k\}$
    Solve $P(b_k + T\hat{y}_{\tau-1}(s), d_k, w_k, C)$. Let $(\hat{x}_\tau(s), \hat{y}_\tau(s), \hat{z}_{\tau+1}(s))$ be the solution.
    Generate Benders cuts from $RB(b_k + T\hat{y}_{t-1}(s), d_k, w_k, \hat{z}_{t+1}(s), C)$.
    Add cuts to C.
  $S^* \leftarrow S^* \cup \tilde{S}_\tau$
  Solve $P(b_0, d_0, w_0, C)$. Let LB be the value of its optimal solution.
  $UB \leftarrow \texttt{ComputeUB}(\tau, S^*, \delta, C, \alpha)$
return $(x_0, y_0), LB, UB$

Algorithm 2 $\texttt{ComputeUB}(\tau, \tilde{S}_\tau, \delta, C, \alpha)$

$U \leftarrow \emptyset$
Determine a steady-state constant state vector $\bar{y}$, see Sect. 4.2.
for $s \in \tilde{S}_\tau$ **do**
  for $t = 0, \ldots, \tau$ **do**
    Solve $P(b_{s_t} + T\hat{y}_{t-1}(s), d_{s_t}, w_{s_t}, C)$. Let $(\hat{x}_t(s), \hat{y}_t(s), \hat{z}_{t+1}(s))$ be the solution.
  $U \leftarrow U \cup \{\sum_{t=0}^{\tau} \delta^t (c^\top \hat{x}_t(s) + h^\top \hat{y}_t(s)) + \delta^{\tau+1} \sum_{k \in K} \text{cost}(\hat{y}_\tau(s), \bar{y}, \omega_k)\}$
return $\text{mean}(U) + \dfrac{q_\alpha}{\sqrt{|\tilde{S}_\tau|}} \text{stdev}(U) + \frac{\delta^{\tau+2}}{1-\delta} \sum_{k \in K} p_k \text{cost}(\bar{y}, \bar{y}, \omega_k)$

is a feasibility problem, consistently with the definition of (MP-Lτ-j).

$$\left.\begin{aligned} \min\ & c^\top x + h^\top y + \delta \sum_{k \in K} p_k z(k) \\ & Ax + Gy \geq \bar{b} \\ & Dx \geq \bar{d} \\ & Wy \geq \bar{w} \\ \forall \gamma \in C \qquad & \gamma_y^\top y + \sum_{k \in K} \gamma_k z(k) \geq \gamma_0 \end{aligned}\right\} \qquad P(\bar{b}, \bar{d}, \bar{w}, C)$$

$$\left.\begin{aligned} \min\ & 0 \\ & Ax + Gy \geq \bar{b} \\ & Dx \geq \bar{d} \\ & Wy \geq \bar{w} \\ & -c^\top x - h^\top y - \delta \sum_{k \in K} p_k z(k) \geq -\bar{z} \\ \forall \gamma \in C \qquad & \gamma_y^\top y + \sum_{k \in K} \gamma_k z(k) \geq \gamma_0 \end{aligned}\right\} \qquad RB(\bar{b}, \bar{d}, \bar{w}, \bar{z}, C)$$

The main idea of the algorithm is to consider a truncated time horizon and apply the sampled nested Benders decomposition scheme to the resulting problem,

dynamically expanding the time horizon until convergence is reached. During the course of the algorithm we take advantage of assumption (A3) and Prop. 1 to immediately add all Benders cuts to all levels of the decomposition. Upper bounds are obtained using a heuristic, see Sect. 4.2, combined with the classical SDDP approach to compute upper bounds via sampling, which would not work on its own because we are considering an infinite-horizon setting.

The B^2 algorithm bears strong similarities with value iteration (see [3,26]), because it iteratively constructs an approximation of the value function: it starts from the empty (i.e., identically zero) approximation, and in each iteration it improves the approximation relying on the one obtained in the previous step. However, a straightforward value iteration scheme would not work in our setting because the state space is infinite, hence we cannot loop over all the states to apply the value function iteration. The convergence rate associated with a value iteration scheme, e.g. [26, Thm. 3.4.2], can in principle be adapted, but its usefulness is limited in our context: its application requires a bound on the absolute change of the value function approximation over the entire space state, which is a polyhedral set in this paper rather than a finite set as in the classical DP setting.

4.2 Computation of upper bounds

Any policy for problem (IHLP) corresponds to an upper bound, but the infinite-horizon nature of the problem introduces difficulties in checking feasibility of a policy, and in computing its value. In practice, to overcome these difficulties we are interested in policies that guarantee feasibility by construction, and for which the corresponding value can be computed in a finite number of steps, e.g. a closed-form formula. The approach that we propose to compute upper bounds is described in Algorithm 2, where the function $\text{cost}(\cdot,\cdot,\cdot)$ and the meaning of the vector $\bar{y}$ will be explained below. The computation is split into two parts: the first part concerns estimating the cost of a feasible policy up to stage τ, the second part computes the cost of a feasible policy from stage τ to ∞. Here, τ is the length of the time horizon considered at that specific iteration by the B^2 algorithm: the first part of the computation is performed in the usual SDDP fashion [24], yielding an upper bound with a given confidence level as explained in [29]. We now discuss the second part of the computation in more detail.

To obtain an estimation of the cost from a given stage τ to ∞, we restrict our attention to *constant-state* policies, defined as policies that always leave the system in the same state regardless of the time index. Note, however, that the existence of such policies is not guaranteed. Therefore, our implementation by default applies a routine that tries to compute a constant-state policy in a heuristic way; if this routine fails, which is possible depending on the problem at hand, it is necessary to provide a problem-specific algorithm to compute upper bounds. The advantage of a constant-state policy is obvious: suppose that given a realization $\omega = \omega_k$, we can compute the cost $\text{cost}(\bar{y}, \bar{y}', \omega_k)$ of a policy that starts from state $\bar{y}$ with right-hand side vector $R_t(\omega_k)$, and leaves the system in state $\bar{y}'$. Then we have:

$$\mathbb{E}_\omega \left[\sum_{t=\tau}^{\infty} \delta^t \text{cost}(\bar{y}, \bar{y}, \omega) \right] = \frac{\delta^\tau}{1-\delta} \left(\sum_{k \in K} p_k \text{cost}(\bar{y}, \bar{y}, \omega_k) \right),$$

because the cost depends only on ω and not on t, as the system is always left in the same state. Provided we have a method to obtain $\text{cost}(\bar{y}, \bar{y}, \omega_k)$, the right-hand side of the equation above is easily computable. We remark that the above equation assumes that the system starts in state $\bar{y}$: the first part of the computation (stage 1 to τ) may not leave the system in state $\bar{y}$, but rather in some state $\hat{y}_\tau$, see Algorithm 2. Thus, we include the term $\text{cost}(\hat{y}_\tau, \bar{y}, \omega_k)$, appropriately discounted, to account for the cost of trasitioning from $\hat{y}_\tau$ to $\bar{y}$, after which we pay $\text{cost}(\bar{y}, \bar{y}, \omega_k)$ at every stage.

Our default strategy to compute the cost of a constant-state policy is the following. We first try to determine a value $\bar{y}$ of the state variables y that allows feasibility to be recovered just by changing the x variables, for all realizations ω_k. In other words, we try to determine a value $\bar{y}$ such that imposing $y_t = \bar{y}$ does not make (IHLP) infeasible. We remark that assumption (A4) does not guarantee that such a value exists because we are now imposing $y_{t-1} = y_t$, which is not part of the original problem. If the desired value $\bar{y}$ exists, we can compute $\text{cost}(\bar{y}, \bar{y}, \omega_k)$ for any value $\bar{y}$ simply by solving (1) setting $y = \bar{y}$ and $V^*(y) \equiv 0$. Hence, the main difficulty is determining $\bar{y}$. We attempt to do so by solving a linear program. Let b_U be the componentwise maximum of $b_1, \ldots, b_\kappa$, d_U the componentwise maximum of $d_1, \ldots, d_\kappa$, w_U the componentwise maximum of $w_1, \ldots, w_\kappa$. We solve the problem:

$$\left.\begin{aligned} \min \quad & c^\top x + h^\top y \\ & Ax + (G - T)y \geq b_U \\ & Dx \geq d_U \\ & Wy \geq w_U, \end{aligned}\right\} \tag{2}$$

that can be interpreted as considering a worst case scenario in which all resources are maximally constrained, $y_{t-1} = y_t$ (hence the constraint $Ax + (G - T)y \geq b_U$), and we try to determine the least-cost state that satisfies the constraints. Note that by definition of b_U, d_U, w_U, if (2) has a solution $\bar{y}$, then we are guaranteed that all scenarios admit a constant-state policy with state $\bar{y}$. If (2) does not have a solution our heuristic fails, and we must resort to a problem-specific approach to construct upper bounds. We remark that based on the problem structure it is sometimes possible to check if (2) is guaranteed to have a solution, e.g. $D \geq 0$, $d_U = 0$, $w_U = 0$, and $A \geq 0$ with $\max_{j=1,\ldots,n_1} A_{ij} > 0$ for all $i = 1, \ldots, m$, in which case $y = 0$ is feasible for (2). This is the type of structure encountered in the two applications tested in Sect. 5. In terms of the production planning example given at the beginning of Sect. 2, the conditions stated above correspond to having unlimited single-stage production capabilities: this implies that the solution with no inventory ($y = 0$) is feasible for (2).

4.3 Convergence

We now discuss convergence of the B^2 algorithm as given in Algorithm 1. The basic idea for the proof is to apply Benders decomposition to the problem obtained with a sufficiently large truncation of the time horizon, and show that as the number of iterations goes to infinity, with probability 1 our algorithm yields value functions that are at least as good as those of the truncated problem. The proof requires some care because the two most natural approaches, stated below, do not seem to work.

The first natural approach would be to apply a standard SDDP convergence proof on the truncated problem. This does not work because such proofs typically use a finiteness argument on the number of cutting planes, i.e. extreme points of the dual problems, that cannot be employed in our case: we keep adding cutting planes to the cut generating problem, therefore the number of extreme points may increase as the iterations progress. The second natural approach would be to use Kelley's cutting plane convergence proof [18] to show convergence to the optimal value function V^*. This does not work because the cutting planes that we generate may not be supporting for V^*. We will eventually rely on the result in [18], but not by applying it directly to V^*.

We call a policy ϵ-*optimal* if, from the initial uncertainty realization R_0, it determines a sequence of actions that yields a total cost no more than ϵ away from the optimal total cost. Our goal is to show convergence to an ϵ-optimal policy. The proof will show that with probability 1 we obtain good value function approximations along any sample path. Notice that our definition of ϵ-optimal policy is only concerned with states that can be visited along a sample path: the proof given below leaves open the possibility that the value function approximation is poor at states that are never visited (i.e., with probability 0) by the ϵ-optimal policy. These states, however, do not affect the expected quality of the policy.

Assumption (A5) implies that it is always possible to identify $\bar{\ell}$ such that the constraint $c^\top x_t(s) + h^\top y_t(s) \geq \bar{\ell}$ is valid (i.e. redundant) for all t. Therefore, $\ell = \sum_{t=0}^{\infty} \delta^t \bar{\ell} = \bar{\ell}/(1-\delta)$ is a lower bound to the total cost of (IHLP). Given ϵ, let $T(\epsilon)$ be the smallest stage index such that

$$\max \left\{ \left| \sum_{t=T(\epsilon)}^{\infty} \delta^t \sum_{s \in S_t} p_s (c^\top x_t(s) + h^\top y_t(s)) \right|, \left| \sum_{t=T(\epsilon)}^{\infty} \delta^t \ell \right| \right\} \leq \epsilon/2, \quad \substack{\forall \text{ feasible} \\ x_t(s), y_t(s),}$$

i.e. $T(\epsilon)$ is large enough that both upper and lower bounds on the cost contribution of subsequent time periods are smaller than $\epsilon/2$ in absolute value. Assumption (A2) together with $0 \leq \delta < 1$ imply that both terms inside the max decrease with $T(\epsilon)$, hence $T(\epsilon)$ exists and is well defined.

Let $P_{T(\epsilon)}$ be the problem obtained by truncating (IHLP) at time horizon $T(\epsilon)$ (included). For stage $t = 1, \ldots, T(\epsilon)$ and $k \in K$, let $V_t^k(y_{t-1})$ be the corresponding value function. Thus, $V_t^k(y_{t-1})$ gives the optimal cost-to-go for $P_{T(\epsilon)}$, i.e. the optimal cost from stage t with state y_{t-1} and scenario $k \in K$ until stage $T(\epsilon)$. Since the time horizon is finite, $P_{T(\epsilon)}$ is a multi-stage linear program, therefore the value functions $V_t^k, t = 1, \ldots, T(\epsilon), k \in K$ exist and are piecewise linear (convex) functions with a finite number of pieces. The optimal policy for $P_{T(\epsilon)}$ can be computed by solving the following problems, where $t = 0, \ldots, T(\epsilon)$ and $j \in K$ (when $t = 0$ take $j = 0$)[1]:

$$\begin{aligned} \min \quad & c^\top x_t + h^\top y_t + \delta \sum_{k \in K} p_k V_{t+1}^k(y_t) & \\ & Ax_t + Gy_t \geq b_j + Ty_{t-1} & \text{(VF-L}t\text{-}j\text{)} \\ & Dx_t \geq d_j & \end{aligned}$$

[1] To be precise, the value of the optimal policy for $P_{T(\epsilon)}$ can be found by solving (VF-L0-0), the remaining problems (VF-Lt-j) are used to identify the optimal policy.

$$Wy_t \geq w_j.$$

By definition of $P_{T(\epsilon)}$ and V_t^k, solving (VF-Lt-j) for $t = 0, \ldots, T(\epsilon)$ following the solutions according to the realization of the sample path is an $\frac{\epsilon}{2}$-optimal policy to (IHLP). Since V_t^k are convex piecewise linear, (VF-Lt-j) can be cast as an LP. We will show that Algorithm 1 constructs an approximation of the value functions V_t^k of $P_{T(\epsilon)}$. We remark that our definition of $T(\epsilon)$ is only important to prove existence of the polyhedral value functions V_t^k; we choose $T(\epsilon)$ so that it yields a total error of $\epsilon/2$ because an additional error of $\epsilon/2$ will be introduced in the value function approximations.

It will be convenient to define $V_t^k(y_{t-1}) = \ell$ for all $t > T(\epsilon)$. This is not restrictive because taking into account the discount factor, the total objective function contribution for stages $t > T(\epsilon)$ is at most $\epsilon/2$. Let n denote the iteration number, and C_n the set of cuts collected up to iteration n. Let $V_{C_n}^k$ be the value function for scenario k induced by the Benders cuts collected in C_n, i.e. $V_{C_n}^k(y) := \min\{z(k) : \gamma_y^\top y + \sum_{j \in K} \gamma_j z(j) \geq \gamma_0 \ \ \forall \gamma \in C_n\}$; thus, $V_{C_n}^k(y_{t-1}) \leq V_t^k(y_{t-1})$ for all $t = 0, \ldots, T(\epsilon)$. We can assume w.l.o.g. that the cut $z(k) \geq \ell$ is implied by C_n for all n and $k \in K$, by initializing C_0 with the cut $z(k) \geq \ell$ if necessary. We call π_n^* the policy implied by the value function approximations obtained from the cuts in C_n. More specifically, the states visited by the policy π_n^* at stages $t \geq 1$ are thus identified by the set of optimal solutions of the problem obtained from (VF-Lt-j) by using the value function approximation $V_{C_n}^k$ rather than V_{t+1}^k. For a given sample path $s \in S_t$, these states are denoted by $\hat{x}_t^s$, $\hat{y}_t^s$, $t = 1, \ldots, T(\epsilon)$.

Define the sets $Q_t^k := \{(y_{t-1}, z_t) : z_t \geq V_t^k(y_{t-1})\}$, i.e. the epigraph of $V_t^k(y_{t-1})$, and $Q_t^k(\bar{\epsilon}) := \{(y_{t-1}, z_t) : z_t \geq V_t^k(y_{t-1}) - \bar{\epsilon}\}$, i.e. an $\bar{\epsilon}$-relaxation of the epigraph of $V_t^k(y_{t-1})$. The next result shows that the sequence of states and the corresponding costs explored by π_n^* converges to Q_t^k for all t.

Lemma 1 *As the number of iterations n of Algorithm 1 goes to infinity, for any $\bar{\epsilon} > 0$, for all $t = 1, \ldots, \infty$, for all sample paths $s \in S_t$, and for all $(\hat{x}_t^s, \hat{y}_t^s)$ visited by following the policy π_n^*, $(\hat{y}_t^s, V_{C_n}^{s_t}(\hat{y}_t^s)) \in Q_t^{s_t}(\bar{\epsilon})$ with probability* 1.

Proof As the number of iterations goes to infinity, the length of the time horizon visited by Algorithm 1 grows infinitely large. Hence, the critical length of the time horizon $T(\epsilon)$ is reached, and we are working with a finite-horizon problem that has $P_{T(\epsilon)}$ as a subproblem.

Define $\hat{\epsilon}_t = \frac{(T(\epsilon)-t)+1}{T(\epsilon)+1}\bar{\epsilon}$ for $t \leq T(\epsilon)$, $\hat{\epsilon}_t = \bar{\epsilon}$ otherwise. We will prove that $(\hat{y}_{t-1}(s), V_{C_n}^{s_t}(\hat{y}_{t-1}(s))) \in Q_t^{s_t}(\hat{\epsilon}_t)$, which implies the original statement since $\hat{\epsilon}_t \leq \bar{\epsilon}$ for all t. Recall that in DP terminology, $\hat{y}_{t-1}^s$ is the endogenous state at time t.

For the sake of contradiction, suppose that there exists a stage index t and a sample path $s \in S_t$ such that $(\hat{y}_{t-1}^s, \hat{z}_t^s = V_{C_n}^{s_t}(\hat{y}_{t-1}^s))$ is visited following π_n^* along sample path s, and $(\hat{y}_{t-1}^s, \hat{z}_t^s) \notin Q_t^{s_t}(\hat{\epsilon}_t)$ with probability greater than zero. Notice that this implies $\hat{z}_t^s < V_t^{s_t}(\hat{y}_{t-1}^s) - \hat{\epsilon}_t$. In case there are multiple sample paths s and stage indices t for which the above statement holds, we consider the sample path s and stage index t such that t is the largest.

First, we claim that $t \leq T(\epsilon)$. Indeed, if $t > T(\epsilon)$, $V_{C_n}^{s_t}(\hat{y}_{t-1}^s) \geq \ell = V_t^k(\hat{y}_{t-1}^s)$, so $(\hat{y}_{t-1}^s, V_{C_n}^{s_t}(\hat{y}_{t-1}^s)) \in Q_t^{s_t} \subset Q_t^{s_t}(\bar{\epsilon})$ for every $\bar{\epsilon} \geq 0$. Thus, we now assume $t \leq T(\epsilon)$.

As $n \to \infty$, a sample path s' that coincides with s up to stage t will be sampled infinitely often. The solution of the forward pass problem for $P(b_{s'_t} + T\hat{y}^{s'}_{t-1}, d_{s'_t}, w_{s'_t}, C_n)$ along s' at stage t is the solution to:

$$\begin{aligned} v := \min \quad & c^\top x^{s'}_t + h^\top y^{s'}_t + \delta \sum_{k \in K} p_k V^k_{C_n}(y^{s'}_t) \\ & Ax^{s'}_t + Gy^{s'}_t \geq b_{s'_t} + T\hat{y}^{s'}_{t-1} \\ & Dx^{s'}_t \geq d_{s'_t} \\ & Wy^{s'}_t \geq w_{s'_t}. \end{aligned}$$

By assumption $V^k_{C_n}(\hat{y}^{s'}_t) \geq V^k_{t+1}(\hat{y}^{s'}_t) - \hat{\epsilon}_{t+1}$ with probability 1 (recall that t is the largest time index $\leq T(\epsilon)$ for which this property does not hold for all sample paths), implying that the objective function value v is greater than or equal to:

$$c^\top \hat{x}^{s'}_t + h^\top \hat{y}^{s'}_t + \delta \sum_{k \in K} p_k V^k_{t+1}(\hat{y}^{s'}_t) - \delta\hat{\epsilon}_{t+1}.$$

Because the first part of the above expression is the cost of performing (potentially suboptimal) action $\hat{x}^{s'}_t$ starting from state $\hat{y}^{s'}_{t-1}$ at stage t, we can write:

$$c^\top \hat{x}^{s'}_t + h^\top \hat{y}^{s'}_t + \delta \sum_{k \in K} p_k V^k_{t+1}(\hat{y}^{s'}_t) - \delta\hat{\epsilon}_{t+1} \geq V^{s'_t}_t(\hat{y}^{s'}_{t-1}) - \delta\hat{\epsilon}_{t+1}.$$

But then, since $\hat{z}^{s'}_t < V^{s'_t}_t(\hat{y}^{s'}_{t-1}) - \hat{\epsilon}_t < V^{s'_t}_t(\hat{y}^{s'}_{t-1}) - \hat{\epsilon}_{t+1} < V^{s'_t}_t(\hat{y}^{s'}_{t-1}) - \delta\hat{\epsilon}_{t+1} \leq v$, a violated Benders cut will be generated during the backward pass. The new cut is either supporting for the set $Q^{s'_t}_t(\hat{\epsilon}_{t+1})$, or cuts inside the set (because v could be strictly greater than $V^{s'_t}_t(\hat{y}^{s'}_{t-1}) - \hat{\epsilon}_{t+1}$). As a straightforward generalization of [18, Sect. 2], the cutting process must converge to a point with distance at most $\frac{1}{T(\epsilon)+1}\bar{\epsilon}$ from the set $Q^{s'_t}_t(\hat{\epsilon}_{t+1})$ after a finite number of cutting planes (the distance $\frac{1}{T(\epsilon)+1}\bar{\epsilon}$ is chosen as the ϵ of the result stated in [18, Sect. 2]). Indeed, notice that the assumptions of [18, Sect. 2] are satisfied, as $\hat{y}^{s'}_{t-1}$ belongs to a compact set, $Q^{s'_t}_t(\hat{\epsilon}_{t+1})$ is convex, and the gradients of the cutting planes are bounded (a cutting plane with unbounded gradient would not be valid for $Q^{s'_0}_0$, a contradiction). Furthermore, $\hat{\epsilon}_{t+1} + \frac{1}{T(\epsilon)+1}\bar{\epsilon} = \hat{\epsilon}_t$. Hence, convergence to a point with distance at most $\frac{1}{T(\epsilon)+1}\bar{\epsilon}$ from the set $Q^{s'_t}_t(\hat{\epsilon}_{t+1})$ implies convergence to a point $(\hat{y}^{s'}_{t-1}, \hat{z}^{s'}_t) \in Q^{s'_t}_t(\hat{\epsilon}_t)$.

To summarize, if a sample path s' coinciding with s up to stage t is sampled infinitely often, then $(\hat{y}^{s'}_{t-1}, \hat{z}^{s'}_t) \in Q^{s'_t}_t(\hat{\epsilon}_t)$ after a finite number of iterations. As $n \to \infty$, this event occurs with probability 1 because each sample path is chosen with positive probability. Thus, the probability that $(\hat{y}^s_{t-1}, V^{s_t}_t(\hat{y}^s_{t-1})) \notin Q^k_t(\hat{\epsilon}_t)$ goes to 0, a contradiction because we had assumed the converse. □

We are now ready to state the main convergence result.

Theorem 1 *For any $\epsilon > 0$, as the number of iterations n of Algorithm 1 goes to infinity, the policy computed by Algorithm 1 converges to an ϵ-optimal policy with probability 1. Furthermore, the algorithm returns a gap that, with an (approximate) confidence level of α, is an upper bound to the gap between the cost of the computed policy, and the cost of the optimal policy.*

Proof The value of the forward pass problem $P(b_0, d_0, w_0, C_n)$ is a lower bound to the cost of the optimal policy by the cut validity argument of Prop. 1. Furthermore, for all $k \in K$ a sample path s with $s_1 = k$ is sampled infinitely often; then, by Lemma 1, the values $V^k_{C_n}(\hat{y}_1(s))$ converge to a $\bar{\epsilon}$-relaxation of the true value functions V^k_1 of the finite-horizon problem $P_{T(\epsilon)}$, for any $\bar{\epsilon} > 0$. Choose $\bar{\epsilon} = \frac{\epsilon}{2\kappa}$. Then $P(b_0, d_0, w_0, C_n)$ underestimates the optimal value of $P_{T(\epsilon)}$ by at most $\epsilon/2$ (each of the value function approximation in the summation of the objective function introduces an error of at most $\frac{\epsilon}{2\kappa}$, and there are κ of them). Recall that by construction, $P_{T(\epsilon)}$ estimates the cost of the optimal policy of (IHLP) with an absolute error of at most $\epsilon/2$. Hence, with probability 1 the lower bound in Algorithm 1 converges to an $(\frac{\epsilon}{2} + \frac{\epsilon}{2}) = \epsilon$-optimal cost, as $n \to \infty$. Lemma 1 shows that the policy π^*_n implied by the value function approximations $V^k_{C_n}$ enjoys the same error bound.

At the same time, as $n \to \infty$, the sample mean of the costs of the (uniformly drawn) sample paths converges to an unbiased estimate of the true cost, and using the central limit theorem, the α quantile of the costs of the sample paths is larger than the true cost up to the current time horizon τ with a confidence of α (the approximation mentioned in the theorem statement is due to the approximation of the distribution of the average with a normal distribution, using the central limit theorem). Since the cost of a heuristic policy from time τ to ∞ provides a deterministic upper bound to the cost from τ to ∞, the upper bound UB computed by Algorithm 2 is a valid upper bound with a confidence of α. When the algorithm stops, the difference between the cost of the policy implicitly defined by the value function approximation, and the cost of the optimal policy, is therefore smaller than $UB - LB$ with an (approximate) confidence of α. □

5 Computational experiments

In this section we report and discuss the results of a computational evaluation of the algorithm proposed in this paper. To test the validity of our algorithm we use a testbed based on two problems considered over an infinite time horizon: Production Planning with Backlog (PPB), and Continuous Rebalancing (CR). For each problem, we provide a mathematical programming formulation, we describe how we constructed the set of instances used in our computational evaluation, and we discuss the empirical behavior of our algorithm, also in comparison to other algorithms when applicable. All the experiments reported in this section are executed on a homogeneous cluster equipped with Xeon E7-4850 processors (2.00GHz 64 GB RAM). Note that since the experiments are run in parallel and job scheduling is unpredictable, the timing statistics reported here are likely to be affected by some noise. However, the ranking of the algorithms indicated by aggregate statistics is so clear that the noise introduced by running parallel computations is irrelevant. In particular, standard deviations can be large and we do not report them for

concisiveness, but our conclusions are supported by additional analysis, as will be explained in the text. Before providing a full description of the test problems, we briefly comment on important implementation choices.

5.1 Implementation details

We implemented the B^2 algorithm in Python 2.7, using Cplex 12.6.1 as an LP solver via its Python interface. As a stopping criterion for the algorithm, we adopted a check on the absolute and relative gap between lower and upper bounds, where the relative gap is computed as $\frac{\text{UB}-\text{LB}}{\text{UB}+10^{-10}}$. Notice that the upper bounds are estimated via sampling and are only valid at a given confidence level δ, set to 95% in our experiments. These stopping criteria are employed with the same tolerances by all algorithms.

The following parameters have a major impact on the computational performance of the B^2 algorithm:

- The number ρ of rounds before purging cuts. We opted for a dynamic management of the set C of Benders cuts, as dynamic management typically improves performance in LP-based cutting plane algorithms. At each iteration, we check which Benders cuts are active. A cut is considered active on a current forward pass solution if the corresponding dual variable is smaller than 10^{-5} in absolute value. If a cut is inactive for more than ρ LP solves, it is removed from C. We remark that in principle this could affect convergence of the algorithm from a theoretical point of view, as the value function approximations may deteriorate after cut removal.
- The number ϕ of sample paths before increasing τ. An important aspects in the B^2 Algorithm is the choice of the size of $\tilde{S}_\tau$, see Alg. 1, line 4: we denote by $\phi = |\tilde{S}_\tau|$ the number of sample paths generated before increasing the length τ of the time horizon.

In the context of SDDP, other cut selection strategies have been discussed. Notably, [8] presents some possibilities (see also [25]) for cut management based on the assumption that cuts should never be deleted to ensure convergence. However, only a subset of the cuts is used at each stage, and an important part of [8] is precisely how to select such a subset. Nevertheless, the difference in performance between all "reasonable" strategies (i.e., excluding the naive approach of keeping a fixed number of cuts in FIFO order) is small. The approach used in this paper is normally used for cut management in MIP solvers, and is known to perform well while introducing little overhead. Of course, care should be taken in ensuring that ρ is not too small, to avoid purging useful cuts just because the sample paths leading to solutions at which those cuts are active are not sampled for a few consecutive iterations.

5.2 Production Planning with Backlog

Production Planning is a classical problem in the stochastic LP literature, see e.g. [27,12] for a discussion on infinite-horizon problems. Production Planning requires deciding the production quantity of a set of items in each stage of the

planning horizon, so as to satisfy demand at minimum cost. In this work, we consider a version of the problem in which backlogs are allowed. Production Planning with Backlogs (PPB) allows postponing part of the demand to a future stage, for a price. Another common PP model allows carrying over inventory to subsequent stages, rather than backlogging. From a mathematical point of view, the two models are similar (the only difference is that some decision variables have their sign flipped), and in our experience they also perform in a similar way from a computational point of view. Since we did not observe any significant computational difference, we limit the discussion to PPB.

The problem can be modeled as follows. We consider the problem of deciding the production effort $x_{j,t}$ dedicated to a certain group of items j at stage t in order to satisfy an uncertain demand. There are M item categories indexed by $i = 1, \ldots, M$ with demands $b_{i,t}$, and N different production plans indexed by $j = 1, \ldots, N$ with the property that dedicating one time unit $x_{j,t}$ to production plan j produces a_{ij} units of item i. Any unfulfilled demand can be backlogged and satisfied in future stages, but this carries a cost. The variable $y_{i,t}$ represents the amount of unsatisfied demand of category i at stage t. We have two costs in the objective function: c_j represents the cost of producing a unit of item j, while h_i represents the cost of backlogging one unit of category i. This yields the following model:

$$\min \sum_{j=1}^{N} c_j x_{j,0} + \sum_{i=1}^{M} h_i y_{i,0} + \sum_{t=1}^{\infty} \delta^t \sum_{k \in K} \Pr(\omega_k)\Big(\sum_{j=1}^{N} c_j x_{j,t} + \sum_{i=1}^{M} h_i y_{i,t}\Big)$$

$$i = 1, \ldots, M \qquad \sum_{j \in N} a_{ij} x_{j,0} + y_{i,0} \geq b_0$$

$$t = 1, \ldots, \infty, i = 1, \ldots, M, k \in K \qquad \sum_{j=1}^{N} a_{ij} x_{j,t} - y_{i,t-1} + y_{i,t} \geq b_{i,t}(\omega_k)$$

$$t = 0, \ldots, \infty, j = 1, \ldots, N \qquad x_{j,t} \geq 0$$

$$t = 0, \ldots, \infty, i = 1, \ldots, M \qquad y_{i,t} \geq 0$$

We generate instances with the following values: $M \in \{10, 20\}$, $N \in \{\frac{M}{2}, M, 2M\}$, and $\kappa \in \{10, 20, 30, 40, 50\}$. Let $U(\alpha, \beta)$ be an integer drawn uniformly at random between α and β. For each combination of the parameters M, N, κ, we generate 10 random instances with the following coefficients: $c_j = R(1, 100)$, $h_i = \dfrac{R(75, 125)}{75N}$, $a_{ij} = \max(0, R(c_j - 10, c_j + 10))$ and $b_i = R(0.75\dfrac{\sum_{j=1}^{N} a_{ij}}{2}, 1.25\dfrac{\sum_{j=1}^{N} a_{ij}}{2})$, where as usual $i = 1, \ldots, M$ and $j = 1, \ldots, N$. This yields a total of 300 PPB instances.

This choice of parameters is dictated by the desire to obtain challenging instances, i.e., instances in which determinining the trade-off between production and backlogging is nontrivial. Our starting point was the discussion in [21] about the generation of difficult knapsack problems.

5.2.1 Experiments on PPB instances

We begin our numerical study with an empirical evaluation of several versions of the B^2 algorithm, as well as the classical SDDP approach. In this section, unless

otherwise stated, the absolute gap is set to 1, δ is set to 0.95, the relative gap is set to 1%, ρ is set to 2 and ϕ is set to 10.

In our first experiment we consider two versions of B^2: a multicut implementation and a single cut implementation. This terminology refers to the way cuts from the scenario subproblems are handled: in a multicut implementation, each violated scenario produces a cut, that is added to the master problem directly; in a single cut implementation, cuts from all scenarios are aggregated based on the corresponding probability, and a single aggregated cut is added to the master problem. As the aggregated cut is implied by the individual cuts, the multicut implementation produces tighter bounds, but this comes at the cost of introducing extra constraints. [7] and [32] investigate the difference between single and multicut. The empirical study of [7] is of limited size (only five problem instances) and it suggests that multicut speeds up convergence. Conversely, [32] shows no clear dominance between single and multicut. Our survey of the literature indicates that single cut implementations are more commonly found than multicut.

Table 1 reports the results of a multicut implementation of B^2 (column "B^2"), a single cut implementation of B^2 (column "B^2 aggr"), and our implemention of the SDDP algorithm [24] (column "PP" for Pereira-Pinto) with multicut for two different values of absolute gap (i.e., 1% and 100%). We remark that all tested algorithms are implemented within the same framework and share most of the subroutines, such as the basic iteration of the forward or backward pass, and the cut management routines. Thus, all algorihms are tested under similar conditions. Unlike the original paper [24], our implementation "PP" is multicut because the computation times for the single cut version are prohibitive. Indeed, Table 1 clearly indicates that B^2 multicut is faster than B^2 single cut: B^2 is faster than B^2 aggr for all instance sizes that we could try. Similarly, preliminary tests indicate that the multicut version of the Pereira-Pinto finite horizon SDDP is faster than single cut, and since the multicut version "PP" is already the slowest algorithm in our numerical evaluation, we only discuss multicut implementations in subsequent sections.

In order to apply finite-horizon SDDP to our problem, given the allowed absolute gap we heuristically determine the length of the time horizon τ' such that the total cost in subsequent time periods does not exceed 90% of the allowed absolute gap. This calculation is based on an estimation of the expected single-stage costs. Once τ' is determined, we apply SDDP employing the usual stopping criteria on absolute and relative gap with a small modification to account for the objective function contribution after τ'.

As remarked earlier, we only report results for small instances, because the running times for PP get prohibitively high on larger problems. While increasing the absolute gap helps especially PP, larger values (e.g., 1000) do not yield a significant reduction of the running times, and a very large absolute gap makes some instances trivial to solve. We verified that B^2 is faster than PP for all instance sizes that we could try. As an additional verification, we performed pairwise comparisons of the algorithms for each group of instances. For equal instance size, B^2 is faster than PP, in all cases. We consider this to be compelling evidence of the ranking of the algorithms.

To understand if the discount factor δ affects the ranking of the algorithms, we report in Figure 1 the average computation time of B^2, B^2 aggr and PP for δ varying from 0.9 to 0.99 (details are reported in Table 5 in Appendix A). The

M	N	k	B^2	B^2 aggr	PP
10	5	10	1.2	7.6	2176.5
		20	5.3	44.6	9376.9
		30	9.2	3843.9	14189.3
		40	11.1	255.4	18392.2
		50	19.7	4021.4	25445.6
Avg.			9.3	1634.6	13916.1

Absolute gap = 1

M	N	k	B^2	B^2 aggr	PP
10	5	10	1.0	5.5	263.8
		20	2.1	22.8	924.6
		30	3.6	45.6	1863.3
		40	5.2	95.4	2361.1
		50	8.8	3739.1	4758.7
Avg.			4.1	781.7	2034.3

Absolute gap = 100

Table 1: Average CPU times for three algorithms on the Production Planning with Backlog problem. Each row indicates the average over 10 instances.

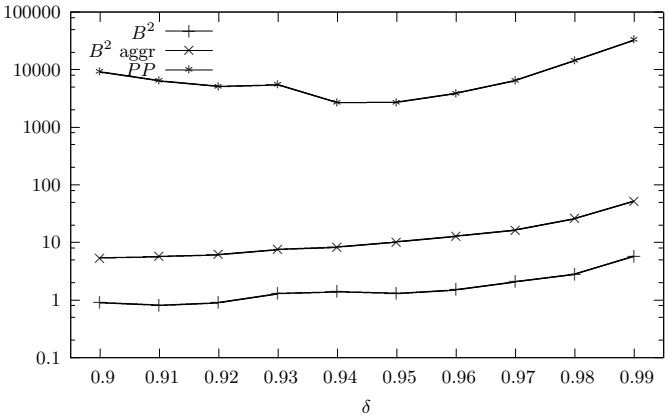

Fig. 1: Computation time (logarithmic scale) vs. values of δ. The instances used in this graph are generated with $M = 10, N = 5, k = 10$. The absolute gap is set to 1.

graph shows that the ranking of the algorithms is not affected by δ, although computation times increase with δ as expected.

Based on this discussion, from now on we focus our attention on B^2 only, with a small absolute gap set to 1 and $\delta = 0.95$.

In an additional round of experiments on PPB instances, we investigate the impact of the algorithmic parameters ρ and ϕ, see Section 5.1, on a large variety of instance sizes. In Table 2 we report average CPU times over 10 instances when using values $\rho = 2, 5, 10$ and $\phi = 2, 5, 10$.

All results obtained with $\rho = 2$ exhibit high standard deviation, indicating unstable behavior (although we remark that for the small instances used in Tables 1 and 5, $\rho = 2$ is sufficient for fast and consistent convergence of B^2). The erratic behavior induced by $\rho = 2$ is evident also from the average times, with large fluctuations across the table. This is due to purging cuts too aggressively, so that useful cuts are frequently deleted and have to be generated again in subsequent iterations. We therefore only analyze results with $\rho = 5$ and $\rho = 10$ in the following. Among the other parameterizations, $\phi = 2$ yields the fastest convergence. Indeed, $\phi = 2$ is faster than $\phi = 5$ and $\phi = 10$ on every instance, not only on average.

			(ρ,ϕ)								
M	N	k	(2,2)	(2,5)	(2,10)	(5,2)	(5,5)	(5,10)	(10,2)	(10,5)	(10,10)
10	5	10	1.4	1.0	1.0	1.4	0.8	1.1	1.6	1.1	1.2
		20	6.9	4.9	5.3	5.2	4.6	5.9	6.1	5.9	5.3
		30	3673.7	7.5	6.2	18.7	8.1	7.2	18.5	10.4	9.2
		40	7.6	6.0	6.6	8.6	7.2	8.6	10.7	9.1	11.1
		50	12.6	13.2	11.5	12.8	12.5	15.2	16.8	16.9	19.7
	10	10	9.8	66.7	76.3	9.6	17.4	42.5	8.9	16.8	35.6
		20	3792.0	3927.4	4574.8	80.9	293.6	846.2	82.9	197.2	541.7
		30	4246.7	1636.8	5712.1	315.6	549.4	2310.1	275.3	563.4	1738.3
		40	3861.4	1681.4	3976.5	160.0	904.5	2177.6	191.0	681.0	1822.9
		50	10407.2	9599.3	13009.1	2479.8	7693.7	9956.6	1289.8	4087.7	9763.9
	20	10	203.4	462.0	1596.3	123.8	353.8	995.7	115.2	306.7	792.0
		20	4389.6	5423.6	14021.8	574.8	2673.5	10796.4	514.4	1838.8	8069.1
		30	4103.2	8671.1	20153.1	1298.2	5835.2	18457.9	1006.5	4661.4	16338.6
		40	15050.6	24742.3	37995.4	4365.7	22362.7	36847.2	2771.4	20072.6	35744.7
		50	16032.6	30212.9	35363.4	13158.7	29841.8	36326.7	12690.2	27620.9	34409.9
20	10	10	4.5	13.0	22.8	4.2	9.4	21.0	4.3	10.1	19.7
		20	52.2	42.2	78.6	33.8	40.8	74.0	38.0	44.5	77.8
		30	316.8	1001.1	1997.6	169.1	560.9	1612.2	150.8	514.7	1376.0
		40	113.4	132.6	278.0	98.2	116.1	161.6	124.3	129.5	220.0
		50	135.5	913.9	3363.5	143.3	672.6	1923.0	178.4	593.6	1759.8
	20	10	17.7	22.0	38.1	15.2	20.7	36.0	16.9	20.9	33.8
		20	357.5	742.3	2277.3	238.2	450.2	1630.2	190.3	424.1	1315.7
		30	572.5	2325.8	7421.4	407.3	1363.6	6065.5	445.8	1197.3	5687.7
		40	6540.4	10881.2	17995.5	1820.5	10200.6	16330.7	1463.5	8499.7	15533.3
		50	8349.3	15674.5	20665.2	6145.2	15511.0	19846.6	5195.7	15418.0	20264.7
	40	10	558.1	1146.8	5627.8	373.3	841.6	3293.8	321.6	726.3	2348.6
		20	15096.6	25070.4	35448.0	4209.2	21087.5	34286.6	2695.4	19003.2	33595.2
		30	29509.0	38800.8	39364.3	23482.8	38752.6	39989.0	20793.4	34792.3	40359.3
		40	29918.5	36610.3	41725.3	28272.9	36621.1	42492.6	27592.6	35925.5	44750.7
		50	38656.2	39688.9	43527.5	36103.6	41466.8	44181.1	36020.7	39861.1	44588.7
Avg.			6533.2	8650.7	11878.0	4137.7	7942.5	11024.6	3807.7	7241.7	10707.8

Table 2: Average CPU times for B^2 with different values of ρ and ϕ on the Production Planning with Backlog problem. Each row indicates the average over 10 instances.

This indicates that increasing the length of the time horizon aggressively is more effective than investing a significant amount of time in exploring many sample paths for fixed time horizon. As expected, increasing k, M, and N makes the problem more difficult to solve. Nevertheless, we are able to solve all instances to within the specified gap.

5.3 Continuous Rebalancing Problem

The Continuous Rebalancing Problem (CRP) concerns the allocation of items from a finite pool to a set of locations, when facing an uncertain demand from customers that move items between two locations, see for example [20]. To give a concrete example, we will call the items "bikes". The locations can then be thought of as bike sharing stations. The system operator can rebalance the number of bikes once

a day (e.g. late at night), incurring some cost. Let N be the set of locations with pairwise distances c_{ij}, $i, j \in N$. Let d_{0i} be the number of bikes at location i at the beginning of the time horizon, and $b_{ij,t}(\omega_k)$ the number of bikes that users wish to rent to go from i to j in period t and scenario k (this number is also referred to as a demand). Let $D = \sum_{i \in N} d_{0i}$ be the total number of bikes. For each location $i \in N$, we define a fictitious location i' that represents location i at the end of a stage; N' is the set of fictitious locations. At the beginning of each time period, $y_{i,t-1}$ represents the number of bikes at location i: the y variables link the stages. Variables $x_{ij,t}$ represent the number of bikes moved from i to j by the system operator at the beginning of time period t, which comes at a cost. Variables $f_{ij',t}$ represent the number of bikes moved by the users during period t ($f_{ii',t}$ represents bikes that did not move, or moved but ended up in their starting location). Finally, variables $u_{ij',t}$ represent lost (unfulfilled) demand between i and j'. We implicitly assume that lost demand has a higher unit cost ℓ_{ij} than the cost for moving a bike incurred by the system operator, otherwise the system operator has no incentive to intervene.

$$\min \sum_{\substack{i,j \in N,\\ j \neq i}} (c_{ij} x_{ij,0} + \ell_{ij} u_{ij,0}) + \sum_{t=1}^{\infty} \delta^t \sum_{k=1}^{\kappa} \Pr(\omega_k) \Big(\sum_{\substack{i,j \in N,\\ j \neq i}} (c_{ij} x_{ij,t} + \ell_{ij} u_{ij,t}) \Big)$$

$$i \in N \qquad \sum_{j \in N} x_{ji,0} - \sum_{j \in N} x_{ij,0} - \sum_{j' \in N'} f_{ij',0} \geq -d_{0i} \tag{3}$$

$$i' \in N' \qquad -y_{i,0} + \sum_{j \in N} f_{ji',0} \geq 0 \tag{4}$$

$$\substack{t=1,\ldots,\infty,\\ i \in N} \qquad \sum_{j \in N} x_{ji,t} - \sum_{j \in N} x_{ij,t} + y_{i,t-1} - \sum_{j' \in N'} f_{ij',t} \geq 0 \tag{5}$$

$$\substack{t=1,\ldots,\infty,\\ i' \in N'} \qquad -y_{i,t} + \sum_{j \in N} f_{ji',t} \geq 0 \tag{6}$$

$$\substack{t=1,\ldots,\infty,\\ k \in K,\\ i \in N, j' \in N'} \qquad f_{ij',t} + u_{ij',t} \geq b_{ij,t}(\omega_k) \tag{7}$$

$$t = 0, \ldots, \infty \qquad \sum_{i \in N, j' \in N'} f_{ij',t} = D \tag{8}$$

$$t = 0, \ldots, \infty \qquad \sum_{i \in N} y_{i,t} \geq D \tag{9}$$

$$\substack{t=0,\ldots,\infty,\\ i,j \in N} \qquad x_{ij,t}, u_{ij,t}, f_{ij',t}, y_{i,t} \geq 0 \tag{10}$$

The objective function minimizes the total cost, that consists of two parts: a cost that the system operator incurs to move bikes in order to rebalance the locations, and a cost for unfulfilled demand. Constraints (3) and (5) represent the natural inventory constraints: users can only take bikes that are present in a station at the beginning of the time period. Constraints (4) and (6) impose that the number of bikes at a given location at the beginning of a time period is equal to the number of bikes brought to that location by users, plus bikes that were already present at the beginning of the time period and did not leave the station. Constraints (7) state that the total demand is split into fulfilled and unfulfilled

demand. Finally (8) and (9) define the total number of bikes that are present in the system.

For the generation of random instances of the CRP problem, besides the size parameter $|N|$ and the number of scenarios κ, we define a saturation parameter ψ that represents the fraction of bikes in the system that are in demand at each stage. If the saturation parameter is small, the system naturally balances itself and the optimal policy is to do nothing. This leads to uninteresting instances from a computational point of view, as the starting policy (with the identically zero function as initial value function estimate) is optimal.

We generated instances with $|N| = 10$, $\kappa \in \{10, 20, 30, 40, 50\}$ and $\psi \in \{1.0, 0.99, 0.98, 0.97\}$. These values for ψ provided the most challenging problems. For each combination of these three parameters, we randomly generate 10 different instances in the following way. First, we sample a base scenario $\bar{b}_{ij} = U(1, 100)$, then set $D = \sum_{i,j \in N, i \neq j} \bar{b}_{ij}$ and $d_{0i} = D/|N|$. We generate the rhs value for the scenarios setting $b_{ij} = U(0.75\psi\bar{b}_{ij},\ 1.25\psi\bar{b}_{ij}), i, j \in N, i \neq j$. We consider only feasible scenarios, i.e., scenarios where the number of bikes is lower than the total demand D. For all instances, we set the cost coefficients to $c_{ij} = 1$ and $\ell_{ij} = 100$, $i, j \in N$. The final CR testbed consists of 200 instances.

5.3.1 Comments on CRP instances

In Table 3, we report results for $\rho = 10, 5$ and $\phi = 2$ on the CR instances.

Solving CR instances of reasonable size to optimality is considerably hard, much harder than PPB instances. This is mainly due to the difficulty of finding a good constant-state policy. Even if the technique explained in Section 4.2 can be applied to CR instances, it typically provides bad primal bounds, and closing the gap often becomes an insurmountable obstacle. For this reason, in Table 3 we restrict ourselves to instances with $N = 10$ and a relative gap (column "gap" in the Table) between 45% and 50%. An interesting observation is that problems with smaller saturation are more difficult to solve: this is because for instances with low saturation there is a large number of unused bikes in each time step, and in extreme cases the optimal policy has a value close to zero (i.e., it is optimal to let the system balance itself). This increases the difficulty because when the lower bound is close to zero, the upper bound also must take a very low value to close the gap, but our primal heuristic makes conservative assumptions on the future demand, and is therefore not able to close the gap.

To investigate in more detail whether or not the difficulty in closing the gap is mainly due to the primal bound, we add an additional stopping criterion based only on the improvement of the dual bound. More precisely, let LB_i be the dual bound obtained at iteration i: we stop the algorithm at iteration i if $\frac{\mathrm{LB}_i - \mathrm{LB}_{i-1}}{\mathrm{LB}_i} \leq 10^{-3}$. In Table 4, we compare the results obtained when adding the new stopping criterion (columns "dual impr") with those obtained with the standard criteria based on absolute and relative gap only (columns "default"). Here we only report results with relative gap 45%, $\rho = 10$ and $\phi = 2$. For each setting, we provide the average CPU time and the final dual bound obtained. We can see that the computation times decreases by at least an order of magnitude with the new stopping criteration, while the dual bound obtained does not change significantly.

			gap (ρ, ϕ)											
			45%		46%		47%		48%		49%		50%	
$\|N\|$	ψ	k	(10,2)	(5,2)	(10,2)	(5,2)	(10,2)	(5,2)	(10,2)	(5,2)	(10,2)	(5,2)	(10,2)	(5,2)
10	0.97	10	2370.7	9155.8	2174.8	5995.0	1683.2	2757.5	1651.5	2316.9	1597.9	1745.6	1375.8	1641.5
		20	2770.3	3242.8	2492.5	3145.8	2439.3	2779.3	2351.8	2689.0	2277.1	2599.0	2233.7	2488.8
		30	5604.5	6960.7	5079.7	6736.3	4857.0	6441.6	4714.1	5719.0	4108.3	5401.5	4038.7	5320.7
		40	9290.5	12701.8	8817.6	12059.7	8564.0	11078.1	8258.1	11058.8	7783.3	10651.0	7533.3	10285.8
		50	22829.6	25433.1	21782.0	25088.1	21602.9	24756.4	20109.5	24384.9	19640.6	24343.2	18977.3	23569.1
	0.98	10	1463.1	1642.0	1368.0	1455.0	1245.7	1416.7	1232.3	1351.9	1138.2	1170.0	1116.6	1142.6
		20	1813.5	2027.2	1837.8	2044.5	1764.6	1951.8	1746.8	1899.0	1610.9	1785.6	1585.1	1747.4
		30	3174.0	3447.5	2897.2	3278.1	2916.6	3200.1	2922.7	3016.2	2811.2	3026.0	2639.9	2840.6
		40	5686.4	6765.8	5314.3	6658.0	5322.3	6480.1	5133.8	6054.9	4765.0	6006.0	4659.5	5799.0
		50	18520.2	22706.6	17757.9	22292.8	17413.2	21652.9	16904.8	21391.9	16028.8	21146.0	15561.2	20696.4
	0.99	10	654.0	762.7	629.3	656.1	661.0	712.1	585.0	693.6	619.0	692.5	585.5	585.3
		20	1232.7	1321.8	1108.7	1190.6	1070.7	1219.8	1112.3	1180.3	1078.5	1075.6	997.2	1106.8
		30	2226.8	2242.3	2179.8	2255.8	2016.6	2170.0	2036.9	2210.6	1960.2	2076.5	1802.3	2057.9
		40	3553.4	3982.9	3312.1	3929.9	3368.0	3852.9	3274.6	3513.7	3193.6	3436.9	3002.3	3470.4
		50	10414.8	14139.6	9597.3	13813.5	9450.4	13526.3	9242.5	12799.0	8682.7	12624.0	8473.5	12212.4
	1.00	10	309.5	344.1	291.0	332.4	308.6	319.7	262.2	307.0	274.6	304.7	261.0	253.0
		20	662.1	702.8	625.9	621.1	623.8	639.8	599.6	612.9	598.9	613.8	550.6	553.3
		30	1331.3	1404.2	1300.7	1305.4	1144.5	1287.0	1138.3	1205.5	1176.0	1127.9	1126.9	1150.8
		40	1944.0	1928.4	1906.7	1917.9	1802.1	1924.7	1752.2	1850.7	1691.4	1745.5	1503.0	1664.6
		50	4379.7	5088.5	3884.7	4905.3	3977.3	4698.4	3829.1	4368.1	3733.0	4292.4	3518.1	4100.8
Avg.			5011.6	6300.0	4717.9	5984.1	4611.6	5643.3	4442.9	5431.2	4238.5	5293.2	4077.1	5134.4

Table 3: Average CPU times for B^2 on the Continuous Rebalancing problem. Each row indicates the average over 10 instances.

			time		dual bound	
N	ψ	k	default	dual impr	default	dual impr
10	0.97	10	2370.7	85.3	4779.9	4429.0
		20	2770.3	86.3	7879.1	7849.8
		30	5604.5	168.8	8004.4	7937.4
		40	9290.5	303.4	7636.2	7614.5
		50	22829.6	741.3	6226.0	6295.5
	0.98	10	1463.1	32.4	6800.5	6648.6
		20	1813.5	47.8	14133.4	14017.9
		30	3174.0	200.7	12943.8	12981.3
		40	5686.4	228.0	12127.5	11834.6
		50	18520.2	347.7	9044.1	9087.4
	0.99	10	654.0	28.7	15628.4	15426.6
		20	1232.7	49.9	26051.2	26286.0
		30	2226.8	162.2	22198.2	21986.6
		40	3553.4	103.7	21060.6	21238.6
		50	10414.8	167.5	17184.5	17521.9
	1.00	10	309.5	20.6	45044.2	45336.9
		20	662.1	28.1	54885.1	55352.5
		30	1331.3	53.2	47497.9	48025.8
		40	1944.0	93.8	48101.6	48646.6
		50	4379.7	138.3	42908.9	43227.0
Avg.			5011.6	154.4	21506.8	21587.2

Table 4: Average CPU times for B^2 on the Continuous Rebalancing problem, using different stopping criteria.

This suggests that a considerable amount of time is spent in trying to improve the primal bound, while the dual bound is already close to its final value.

6 Conclusions

This paper presents a provably convergent nested decomposition algorithm for infinite-horizon stochastic LP. The convergence proof is based on a combination of Benders decomposition with Kelley's cutting plane algorithm. The advantage of the approach proposed in this paper is that it allows us to use the well-known Benders scheme on an infinite-horizon problem, while retaining strong theoretical guarantees.

Our computational experiments lead to the following conclusions. First, the multicut implementation in the nested decomposition scheme converges much faster than a single cut implementation. The effect is particularly evident in our setting because our algorithm relies heavily on Benders decomposition, in the sense that all subproblems in the decomposition scheme are solved by Benders decomposition themselves. This compounds the effect of implementation details, and shows that keeping a cut for each violated scenario is worth the computational burden, thanks to tighter bounds. Second, the construction of primal bounds is a delicate step in the infinite-horizon setting: it can lead to a deterioration of performance if general-purpose heuristics are not effective and problem-specific primal bounds are not available. However, in our experiments the dual bound always converges

relatively quickly, so that even if we cannot prove tight gap guarantees, the set of cuts generated by the algorithm should yield high-quality policies.

References

1. Asamov, T., Powell, W.B.: Regularized decomposition of high-dimensional multistage stochastic programs with markov uncertainty. SIAM Journal on Optimization **28**(1), 575–595 (2018)
2. Bellman, R.: Dynamic Programming. Princeton University Press, Princeton, NJ (1957)
3. Bertsekas, D.P.: Dynamic Programming and Optimal Control. Athena Scientific, Belmont, MA (1995)
4. Bertsimas, D., Tsitsiklis, J.: Introduction to Linear Optimization. Athena Scientific, Belmont (1997)
5. Birge, J.R.: Decomposition and partitioning methods for multistage stochastic linear programs. Operations Research **33**(5), 989–1007 (1985)
6. Birge, J.R., Louveaux, F.: Introduction to stochastic programming. Springer Science & Business Media (2011)
7. Birge, J.R., Louveaux, F.V.: A multicut algorithm for two-stage stochastic linear programs. European Journal of Operational Research **34**(3), 384 – 392 (1988)
8. de Matos, V.L., Philpott, A.B., Finardi, E.C.: Improving the performance of stochastic dual dynamic programming. Journal of Computational and Applied Mathematics **290**, 196 – 208 (2015)
9. Donohue, C., Birge, J.R.: The abridged nested decomposition method for multistage stochastic linear programs with relatively complete recourse. Algorithmic Operations Research **1**(1) (2006)
10. Duffin, R.J., Karlovitz, L.A.: An infinite linear program with a duality gap. Management Science **12**(1), 122–134 (1965)
11. Gassmann, H.I.: Mslip: A computer code for the multistage stochastic linear programming problem. Mathematical Programming **47**(1), 407–423 (1990)
12. Ghate, A., Smith, R.: Optimal backlogging over an infinite horizon under time-varying convex production and inventory costs. Manufacturing & Service Operations Management **11**(2), 362–368 (2009)
13. Girardeau, P., Leclere, V., Philpott, A.B.: On the convergence of decomposition methods for multistage stochastic convex programs. Mathematics of Operations Research **40**(1), 130–145 (2014)
14. Grinold, R.C.: Finite horizon approximations of infinite horizon linear programs. Mathematical Programming **12**(1), 1–17 (1977)
15. Guigues, V.: SDDP for some interstage dependent risk-averse problems and application to hydro-thermal planning. Computational Optimization and Applications **57**(1), 167–203 (2014)
16. Heitsch, H., Römisch, W.: Scenario tree modeling for multistage stochastic programs. Mathematical Programming **118**(2), 371–406 (2009)
17. Infanger, G., Morton, D.P.: Cut sharing for multistage stochastic linear programs with interstage dependency. Mathematical Programming **75**(2), 241–256 (1996)
18. Kelley, J.E.: The cutting-plane method for solving convex programs. Journal of the Society of Industrial and Applied Mathematics **8**(4), 703–712 (1960)
19. Kim, K., Mehrotra, S.: A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management. Operations Research **63**(6), 1431–1451 (2015)
20. Laporte, G., Meunier, F., Wolfler Calvo, R.: Shared mobility systems. 4OR **13**(4), 341–360 (2015)
21. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, Inc., New York, NY, USA (1990)
22. Mulvey, J.M., Vladimirou, H.: Solving multistage stochastic networks: an application of scenario aggregation. Networks **21**(6), 619–643 (1991)
23. Mulvey, J.M., Vladimirou, H.: Stochastic network programming for financial planning problems. Management Science **38**(11), 1642–1664 (1992)
24. Pereira, M., Pinto, L.: Multi-stage stochastic optimization applied to energy planning. Mathematical Programming **52**(1-3), 359–375 (1991)

25. Pfeiffer, L., Apparigliato, R., Auchapt, S.: Two methods of pruning Benders' cuts and their application to the management of a gas portfolio. Tech. Rep. 8133, INRIA (2012)
26. Powell, W.B.: Approximate Dynamic Programming: Solving the Curses of Dimensionality, 2nd Edition. Wiley (2011)
27. Romeijn, E.H., Smith, R.L., Bean, J.C.: Duality in infinite dimensional linear programming. Mathematical Programming **53**(1-3), 79–97 (1992)
28. Schochetman, I.E., Smith, R.L.: Finite dimensional approximation in infinite dimensional mathematical programming. Mathematical Programming **54**(1-3), 307–333 (1992)
29. Shapiro, A.: Analysis of stochastic dual dynamic programming method. European Journal of Operational Research **209**(1), 63 – 72 (2011)
30. Van Slyke, R.M., Wets, R.: L-shaped linear programs with applications to optimal control and stochastic programming. SIAM Journal on Applied Mathematics **17**(4), 638–663 (1969)
31. Zenios, S.A.: Financial Optimization. Cambridge University Press, Cambridge, UK (1993)
32. Zhang, W.: Water network design and management via stochastic programming. The University of Arizona (2013)
33. Zou, J., Ahmed, S., Sun, X.A.: Nested decomposition of multistage stochastic integer programs with binary state variables. Tech. rep., Georgia Institute of Technology (2016). URL `http://www.optimization-online.org/DB\_HTML/2016/05/5436.html`

A Computational tests with different values of δ

In Table 5 we show the details about the computation times of the different algorithms when δ increases.

δ	M	N	k	B^2	B^2 aggr	PP
0.90	10	5	10	0.9	5.4	9137.2
			20	7.0	34.1	8182.4
			30	8.6	3729.1	6525.6
			40	10.2	165.5	19246.7
			50	20.7	3838.5	15689.9
	Avg.			9.5	1554.5	11756.3
0.91	10	5	10	0.8	5.7	6339.6
			20	6.8	36.3	8085.4
			30	8.6	3753.6	7685.9
			40	10.2	171.7	18519.8
			50	17.6	3815.6	16159.8
	Avg.			8.8	1556.6	11358.1
0.92	10	5	10	0.9	6.2	5093.6
			20	6.0	39.7	6995.6
			30	8.4	3770.2	8900.6
			40	10.0	179.4	17885.9
			50	17.3	3837.9	18451.4
	Avg.			8.5	1566.7	11465.4
0.93	10	5	10	1.3	7.6	5452.5
			20	6.4	44.4	6510.4
			30	9.4	3780.9	10732.8
			40	8.2	206.5	13604.5
			50	20.0	3855.1	19915.5
	Avg.			9.1	1578.9	11243.1
0.94	10	5	10	1.4	8.3	2683.0
			20	5.8	49.0	7225.2
			30	10.2	3840.7	10636.9
			40	11.9	208.8	16873.6
			50	24.1	3920.5	22571.4
	Avg.			10.7	1605.4	11998.0
0.95	10	5	10	1.3	10.3	2710.3
			20	6.1	54.6	9670.6
			30	9.8	3819.2	16110.7
			40	11.9	270.3	20278.3
			50	21.8	3973.7	26071.2
	Avg.			10.2	1625.6	14968.2
0.96	10	5	10	1.5	12.8	3861.6
			20	8.6	64.8	10956.1
			30	9.7	3846.9	19455.9
			40	11.8	333.0	25230.4
			50	17.3	4099.4	28982.2
	Avg.			9.8	1671.4	17697.2
0.97	10	5	10	2.1	16.5	6555.8
			20	9.8	84.0	16924.4
			30	13.4	3950.8	28367.6
			40	14.3	407.7	31461.0
			50	26.5	4202.1	31953.4
	Avg.			13.2	1732.2	23052.4
0.98	10	5	10	2.8	26.0	14520.0
			20	12.9	127.0	26234.7
			30	17.8	4168.4	36000.0
			40	21.0	592.3	36000.0
			50	32.1	4436.7	35679.0
	Avg.			17.3	1870.1	29686.8
0.99	10	5	10	5.8	52.3	32586.6
			20	32.4	271.9	35305.7
			30	43.7	3019.6	36000.0
			40	48.1	1235.7	36000.0
			50	93.0	5255.0	36000.0
	Avg.			44.6	1966.9	35178.4

Absolute gap = 1

δ	M	N	k	B^2	B^2 aggr	PP
0.90	10	5	10	0.8	5.2	9132.1
			20	3.7	23.4	8505.6
			30	5.7	57.1	7299.8
			40	7.0	83.4	20005.0
			50	15.0	169.9	15791.8
	Avg.			6.4	67.8	12146.8
0.91	10	5	10	0.8	5.4	5505.9
			20	3.3	25.1	8541.8
			30	5.7	58.4	8576.3
			40	7.7	86.3	18931.2
			50	12.4	138.5	15113.0
	Avg.			6.0	62.8	11333.6
0.92	10	5	10	0.9	6.4	5101.0
			20	3.8	27.6	7766.2
			30	5.6	69.5	8946.8
			40	7.7	96.3	16383.8
			50	11.8	228.6	18964.8
	Avg.			6.0	85.7	11432.5
0.93	10	5	10	1.3	7.5	5382.5
			20	4.8	30.9	7421.1
			30	7.5	78.9	10747.2
			40	7.3	113.0	16037.6
			50	18.0	261.9	18641.6
	Avg.			7.8	98.5	11646.0
0.94	10	5	10	1.3	8.2	2622.9
			20	4.9	36.1	8332.1
			30	8.4	91.5	13362.8
			40	10.7	122.7	15189.4
			50	21.5	274.2	23230.0
	Avg.			9.4	106.5	12547.4
0.95	10	5	10	0.9	9.9	2707.5
			20	5.0	43.0	10695.0
			30	6.6	112.3	16244.9
			40	12.3	154.3	20670.2
			50	13.7	382.2	26082.5
	Avg.			7.7	140.3	15280.0
0.96	10	5	10	1.5	12.7	3915.3
			20	5.2	55.5	10025.2
			30	10.1	138.8	20867.9
			40	12.1	245.7	22452.7
			50	18.9	502.9	29430.4
	Avg.			9.6	191.1	17338.3
0.97	10	5	10	2.1	16.4	6501.5
			20	6.6	73.3	17165.8
			30	12.4	187.1	25055.3
			40	14.8	306.7	32520.1
			50	28.9	646.3	31310.2
	Avg.			13.0	246.0	22510.6
0.98	10	5	10	2.8	26.1	16395.5
			20	9.5	110.9	23445.6
			30	17.0	281.5	34534.8
			40	22.2	502.9	36000.0
			50	34.4	997.5	35213.4
	Avg.			17.2	383.8	29117.9
0.99	10	5	10	5.7	52.0	29514.9
			20	19.8	225.9	35215.0
			30	35.4	565.4	36000.0
			40	51.0	1053.1	36000.0
			50	98.0	2097.9	36000.0
	Avg.			42.0	798.9	34546.0

Absolute gap = 100

Table 5: Extended results concerning the average CPU times for three algorithms on the Production Planning with Backlog problem. Each row indicates the average over 10 instances.