The policy graph decomposition of multistage stochastic optimization problems

Oscar Dowson

Correspondence
Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, 60208, IL
Email: os-car.dowson@northwestern.edu

Funding information

We propose the policy graph as a way of formulating multistage stochastic optimization problems. We also propose an extension to the stochastic dual dynamic programming algorithm to solve a class of problems formulated as a policy graph. This class includes discrete-time, infinite horizon, multistage stochastic optimization problems with continuous state and control variables.

Keywords
policy graph, multistage, stochastic programming, infinite horizon

1 | INTRODUCTION

This paper presents a new way of formulating and solving a general class of multistage stochastic optimization problems. Its most notable contribution is an algorithm for solving discrete-time, infinite horizon stochastic dynamic programming problems with continuous state and control variables. Our motivation is threefold: (1) to provide a solution algorithm for the infinite horizon case; (2) to generalize many of existing special case formulations of multistage stochastic optimization models in the literature; and (3) to provide a new notation that allows the practitioner to cleanly communicate a much broader class of multistage stochastic optimization models.

Unlike the standardization that has taken place in deterministic optimization (where terms like decision variable, constraint, and objective function are widely accepted), the stochastic optimization community has fragmented into different communities, each of which speaks a different arcane language (Powell, 2014). Therefore, in order to introduce our problem setting and explain our solution technique, it is necessary to clearly define the terminology and notation that we shall be using in this paper. Readers should be aware that compared with other approaches in the literature, there are some subtle (and some large) differences in our approach. We use the language of stochastic optimal control (Bertsekas, 2005), with terms like stage, state, and control. We do not view the world as a scenario tree. Nor do we explicitly express the uncertainty in terms of a filtration on some probability space. Instead, we follow an approach that is heavily inspired by the approximate dynamic programming framework of Powell (2011) and the Markov decision processes of Puterman (1994): we decompose the problem into small atomic pieces, clearly define each piece, and then
define how the pieces fit together.

The paper is laid out as follows. In Section 2, we begin with some necessary terminology. Then, in Section 3, we introduce the policy graph as a way of formulating multistage stochastic optimization problems. In Section 4, we present an algorithm to solve a subset of problems formulated as a policy graph. Finally, in Section 5 we demonstrate the utility of considering the infinite horizon formulation of a problem arising from pastoral agriculture.

2 | TERMINOLOGY

First, let us define the stage in multistage.

**Definition** A stage is a discrete moment in time in which the agent chooses a decision and any uncertainty is revealed.

Therefore, multistage refers to a problem that can be decomposed into a sequence of stages. This requires the assumption that time can be discretised. Second, the stochastic component of multistage stochastic optimization refers to problems with uncertainty. In this paper, we differentiate between two types of uncertainty; the first of which we term a noise. (We shall describe the second type in the next section, but it relates to how the problem transitions between stages.)

**Definition** A noise is a stagewise-independent random variable in stage $t$.

In stage $t$, we denote a single observation of the noise with the lowercase $\omega_t$, and the sample space from which it is drawn by the uppercase $\Omega_t$. $\Omega_t$ can be continuous or discrete, although in this paper we only consider the discrete case. Furthermore, we use the term stagewise-independent to refer to the fact that the noise in stage $t$ is sampled independently of the noise terms in stages $1, 2, \ldots, t-1, t+1, t+2, \ldots$.

Next we define a state, modifying slightly the definition from Powell (2016).

**Definition** A state is a function of history that captures all the information we need to model a system from some point in time onward.

Expressed a different way, a state is the smallest piece of information that is necessary to pass between stage $t$ and $t+1$ so that the optimal decision-making in stage $t+1$ onward can be made independently from the decisions that were made in stages 1 to $t$. Each dimension of the state is represented by a state variable. State variables can be continuous or discrete.

We denote the state variable at the start of stage $t$ by the lowercase $x_t$. We refer to $x_t$ as the incoming state variable. Then, during the stage: the agent chooses a control (action); a realization of the noise is observed; and the state transitions to $x_t'$ at the end of the stage according to the transition function. We refer to $x_t'$ as the outgoing state variable. We now define a control and the transition function.

**Definition** A control variable is an action or decision taken (explicitly or implicitly) by the agent during a stage.

Control variables in stage $t$ can be continuous or discrete. In this paper, we denote control variables with the lowercase $u_t$. Such controls must be feasible for the agent to implement, and therefore they belong to a set that depends on the incoming state variable and observation of the random noise; this is denoted $u_t \in U_t(x_t, \omega_t)$.

**Definition** The transition function is a mapping of the incoming state $x_t$ to the outgoing state $x_t'$, given the control $u_t$ and the noise $\omega_t$. 

We denote the transition function as \( x_t' = T_t(x_t, u_t, \omega_t) \). This function can be of any form. As a result of the state transitioning, a cost is incurred.

**Definition** The stage-objective is the cost (if minimizing, otherwise value) accrued in stage \( t \) as a consequence of taking the control \( u_t \), given the incoming state \( x_t \) and realization of the noise \( \omega_t \).

This is denoted \( C_t(x_t, u_t, \omega_t) \). All that remains is to define how the agent chooses a control. We use the terminology of Puterman (1994) and call this a decision-rule.

**Definition** A decision-rule \( \pi_t \), for stage \( t \), is a mapping of the incoming state variable \( x_t \) and observation of the noise \( \omega_t \) to a control \( u_t \).

This is denoted \( u_t = \pi_t(x_t, \omega_t) \). In cases where the decision-rule does not depend upon the noise, we denote the decision-rule as \( \pi_t(x_t) \). We refer to a set of decision-rules, one for each stage \( t \), as a policy.

**Definition** A policy is a set of decision-rules \( \pi = \{ \pi_t : t = 1, 2, \ldots, T \} \), containing one element for each stage \( t \).

Now that we have defined some basic terminology, we can construct the atomic building block of multistage stochastic optimization, which we refer to as a node.

**Definition** A node is a collection of the following components: incoming and outgoing state variables, a noise, some control variables, a transition function, a stage-objective, and a decision-rule.

In this paper, we discriminate between two types of node:

1. a Hazard-Decision node (also called Wait-and-See (Birge and Louveaux, 2011)); and
2. a Decision-Hazard node (also called Here-and-Now (Birge and Louveaux, 2011)).

We now define each type of node in turn.

**Definition** In a Hazard-Decision node, the agent chooses a control \( u_t \) after observing a realization of the noise \( \omega_t \in \Omega_t \) according to the decision-rule \( \pi_t(x_t, \omega_t) \). The state transitions from \( x_t \) to \( x_t' \) according to the transition function \( T_t(x_t, u_t, \omega_t) \). The decision-rule respects the set of admissible controls so that \( u_t \in U_t(x_t, \omega_t) \). In addition, a cost \( C_t(x_t, u_t, \omega_t) \) is incurred. A schematic of this is shown in Fig. 1.

![Schematic of a Hazard-Decision node](image-url)

**FIGURE 1** Schematic of a Hazard-Decision node.
**Definition** In a Decision-Hazard node, the agent chooses a control $u_t$ before observing a realization of the noise $\omega_t \in \Omega_t$ according to the decision-rule $\pi_t(x_t)$. The state transitions from $x_t$ to $x'_t$ according to the transition function $T_t(x_t, u_t, \omega_t)$. The decision-rule respects the set of admissible controls so that $u_t \in U_t(x_t)$. In addition, a cost $C_t(x_t, u_t, \omega_t)$ is incurred. A schematic of this is shown in Fig. 2.

When no uncertainty is realized in the node (i.e., $|\Omega_t| = 1$), the Decision-Hazard node becomes identical to the Hazard-Decision node. We denote this deterministic node by dropping the wavy line coming into the box depicting the node and dropping the $\omega_t$ function arguments. Furthermore, readers should note that the two types of nodes are not immutable. Instead, they are a modelling choice. For example, it is possible to transform a Decision-Hazard node into a deterministic node followed by a Hazard-Decision node with an expanded state-space. An example of this is shown in Fig. 3. Note that we pass $x_t$ and $u_t$ between the nodes, instead of just $x_t$. Despite the fact that this example demonstrates that we can remove the concept of a Decision-Hazard node, we retain the distinction between Decision-Hazard and Hazard-Decision because it is a useful modelling device.

**FIGURE 2** Schematic of a Decision-Hazard node.

**FIGURE 3** A Decision-Hazard node expanded into a deterministic node and a Hazard-Decision node.

The definitions above should be familiar to most readers versed in stochastic optimization. However, notably excluded is a description of how the nodes are linked together. Typically, nodes are linked together as a linear sequence so that $x'_t = x_{t+1}$. In this case, we use the terms node and stage interchangeably, and we use $x_t$ and $x_{t+1}$ instead of $x_t$ and $x'_t$. Implicit within this formulation is the idea that $u_t$ cannot depend upon events in stages $t + 1$ onward (i.e., nonanticipativity), since the decision-rule $\pi_t$ depends only upon the incoming state variable $x_t$ and the stagewise independent random variable $\omega_t$.

In contrast to the linear case, it is possible to link nodes together in more complicated structures (e.g., such as the Markovian lattice used by Philpott and de Matos (2012)). In this case, there can be many nodes corresponding to each stage (in time). Therefore, we cannot use the terms node and stage interchangeably. To clearly signal the difference, we will use the subscript $i$ instead of $t$ when referring to the components within each node. Moreover, for each of the previous definitions, the reader should replace all references to “stage $t$” with “node $i$.” Additionally, we really
mean “nodewise-independent” instead of “stagewise-independent,” but we shall retain the terminology of stagewise independence. In the next section, we generalize this idea of linking nodes with what we term a policy graph.

3 | POLICY GRAPHS

In the previous section, we defined two types of node: Hazard-Decision and Decision-Hazard. In this section, we describe how to link nodes together. Thus, we first need to define the initial conditions of the problem.

**Definition** The root node is the current point-of-view of the agent in the decision-making process and stores an initial value $x_R$ for the state variable.

It is important to note that the root node is neither a Hazard-Decision nor a Decision-Hazard node; it is just a convenient object to represent the initial point in the sequential decision-making process.

We now have all the terminology necessary to define the object that specifies the structure of how the nodes link together. We call the structure a policy graph.

**Definition** A policy graph $G = (R, N, E, \Phi)$ is defined by a tuple containing the root node $R$, along with the set of nodes $N$ and directed edges $E$. $\Phi$ is an $|N| + 1$ by $|N|$ matrix of the transition probabilities between nodes, with entries $\phi_{i,j}$, such that if there exists an edge $(i, j)$ in $E$ for two nodes $i \in N \cup \{R\}$ and $j \in N$, then $x_j = x_i'$ with probability $\phi_{i,j} \geq 0$. If no edge exists, then $\phi_{i,j} = 0$.

This definition encompasses the second type of uncertainty alluded to in the previous section: the probability of transitioning between nodes in the policy graph. (Recall the first type of uncertainty was the stagewise-independent noise within a node.) Before we elaborate on the transitions between nodes, it is useful to define the children of a node and the notion of two nodes being connected.

**Definition** The **children** of node $i$ are the nodes in the set $i^+ = \{j : \phi_{i,j} > 0\}$.

**Definition** Node $i$ is **connected** to node $j$ if $j \in i^+$.

**Definition** Node $i$ is a **leaf** node if $i^+ = \emptyset$.

In the remainder of this section we show, by graphical example, the various features of a policy graph. The graphical representation of each policy graph has a single circle that represents the root node. The square boxes represent Hazard-Decision or Decision-Hazard nodes. For simplicity, we drop the annotations used in Figures 1 and 2, since the node type can be inferred based on the presence (or absence) of a wavy arc representing the noise.

We note that this paper is not the first to advocate for a graphical representation of the decision-making process. The System Dynamics community have developed a rich graphical framework for conveying dynamical systems (Sterman, 2000). However, the System Dynamics approach focuses on the (causal) relationships between the states (stocks) and controls (flows) and cannot express how uncertainty is revealed through the decision-making process. There are also well-known tools in the Operations Research community that use similar symbols such as decision trees, scenario trees, and flowcharts. (See Powell (2011) for examples.) Our approach is different in that it operates at a level of abstraction above the System Dynamics framework (for example, each node can be diagrammed using System Dynamics), and that (as we shall show) it supersedes the scenario tree approach.
Sequential Hazard-Decision problem

In a sequential Hazard-Decision problem, the noise is observed at each square node before the control is chosen. A graphical representation of the policy graph is given in Fig. 4. Note that the arcs can be interpreted as the flow of information (i.e. the states) between nodes.

The node graph is:

\[ G = (R, \{N, E, \Phi \}) = (R, \{B, C\}, \{(R, B), (R, C), (B, D), (B, E), (C, F), (C, G)\}), \{\phi_{R,B} = 0.5, \phi_{R,C} = 0.5, \phi_{B,D} = 0.5, \phi_{B,E} = 0.5, \phi_{C,F} = 0.5, \phi_{C,G} = 0.5 \}). \]

We call a policy graph of this form a linear policy graph.

**Definition** A linear policy graph is composed of a finite set of nodes, where each node is connected to, at most, one other node.

Recall that when the policy graph is linear, we can use the terms node and stage interchangeably, and we denote the outgoing state variable in stage \( t \) as \( x_{t+1} \) instead of \( x_t' \).

A policy graph can also be thought of as a compressed scenario tree. If each Hazard-Decision node in Fig. 4 has two possible realizations for \( \omega_t \), then the scenario tree has six nodes. This can be described by the policy graph:

\[ G = (R, N, E, \Phi) = ( \]
\[ R, \]
\[ \{B, C, D, E, F, G\}, \]
\[ \{(R, B), (R, C), (B, D), (B, E), (C, F), (C, G)\}\],
\[ \{\phi_{R,B} = 0.5, \phi_{R,C} = 0.5, \phi_{B,D} = 0.5, \phi_{B,E} = 0.5, \phi_{C,F} = 0.5, \phi_{C,G} = 0.5 \}). \]

The graphical representation is shown in Fig. 5.

The policy graphs in Figures 4 and 5 are equivalent representations of the same problem. However, the first example contains more information than the second example since it shows that node D is identical to node F, and node E is identical to node G. To clarify the point, any policy graph (with a finite sample space for the noise \( \Omega \)) can be expanded into a scenario tree, but a scenario tree cannot be compressed into a linear policy graph without the additional information.
of which transitions between nodes are stagewise-independent.

The benefit of the graphical representation of a policy graph is most apparent when different types of nodes are mixed together. For example, a common problem in the literature is the two-stage problem with recourse (Fig. 6). In this problem, the first stage is deterministic, and the second stage is Hazard-Decision. Alternatively, it is possible to formulate the problem as shown in Fig. 7: the first stage is Decision-Hazard, and the second stage is deterministic.

**FIGURE 6** Two-stage problem with recourse: hazard-decision.

**FIGURE 7** Two-stage problem with recourse: decision-hazard.

### 3.1 Conditional dependence

So far, we have limited the uncertainty to two types: (1) a stagewise-independent noise within a node; and (2) probabilistic transitions between nodes. Importantly, the transition between the nodes determines the sample space from which the stagewise-independent noise is sampled. We now show how these two types of uncertainty can be combined to produce stagewise-dependent\(^1\) noise. As a motivating example, consider a hypothetical ice-cream seller in New Zealand. The New Zealand summer climate is dominated by the El Niño-Southern Oscillation, an irregular, periodical climate pattern in the Pacific Ocean. It has two extremes: El Niño and La Niña. In El Niño years, the Eastern Pacific warms relative to average, and there is less rainfall than average in New Zealand. In La Niña years, the opposite is true (NIWA, 2018). Prior to learning if the year is El Niño or La Niña, an ice-cream seller must order a quantity of ice-cream to sell over the summer. Then, during the summer, their sales depend upon the number of sunny days. The number of sunny days is uncertain, although there is a higher probability of sunny days during El Niño years than La Niña.

We could model this problem by a linear, two-stage policy graph like Fig. 6. However, we would need to add a binary state variable to code whether the system was in El Niño or La Niña. This formulation comprises one decision-rule for the first stage and one decision-rule for the second stage.

Alternatively, we can model the problem as shown in Fig. 8 without the need for the additional state variable. This formulation comprises one decision-rule for the first stage (as before), but two decision-rules for the second stage: one to use if the year is El Niño and one to use if the year is La Niña.

**FIGURE 8** Two-stage problem with recourse and conditional dependence.

---
\(^1\)We really mean node-wise-dependent, but this may confuse the matter too much.
Importantly, the realization of the random node transition (into either El Niño or La Niña) is exogenous to the remainder of the problem. Such a decomposition is hardly novel – indeed it is identical to the process of decomposing the problem into stages (and thereby solving for a decision-rule in each stage, instead of a single decision-rule that includes time as a state variable).

If we extend this structure to a series of stages, where all the nodes in a stage $t$ are connected to all the nodes in the next stage $t + 1$, we end up with a Markov chain structure. We call policy graphs with this structure Markovian.

**Definition** A Markovian policy graph is composed of a finite set of stages, where each stage consists of a set of nodes, and all the nodes within a stage are connected to nodes within, at most, one other stage.

An example of a Markovian policy graph is shown in Fig. 9. We refer to the different nodes within a stage as the Markov states of the stage. Note that a linear policy graph is a Markovian policy graph with one Markov state in each stage.

![Figure 9](image)

**FIGURE 9** A Markovian policy graph.

The scenario tree in Fig. 5 is also a Markovian policy graph. However, there are some structures which are neither linear nor Markovian; an example of which is shown in Fig. 10. In that example, the policy graph is not Markovian as the node in the first stage has one child in the second stage and one in the third stage. Variations of these acyclic policy graphs are well studied in the work of Rebennack (2016).

![Figure 10](image)

**FIGURE 10** A non-Markovian policy graph.

Non-Markovian policy graphs reinforce the need to distinguish between a node and a stage. However, once this distinction is made, we can easily express problems with a random number of stages such as those considered by Guigues (2018). An example is shown in Fig. 11. In this policy graph, it appears that there are a random number of stages because the agent may make 2, 3, or 4 decisions depending on which path they take through the graph. This example is a good demonstration of how the policy graph approach is able to generalize existing special cases in the literature.

### 3.2 Cyclic policy graphs

The examples above were all acyclic. However, the policy graph approach naturally extends to infinite-horizon problems. For example, Fig. 12 shows an infinite-horizon, Decision-Hazard problem with some probability of exiting the cycle into
a final recourse node.

One approach to solving infinite-horizon problems is to incorporate a discount factor in order to produce a finite sum of the future costs (assuming, of course, a bounded stage-objective); this can be represented in a policy graph by allowing the probabilities of the arcs exiting a stage to sum to less than one. Implicitly, this results in the addition of a leaf node with zero cost (Fig. 13). The discount factor $p$ can be interpreted as a $1 - p$ probability that the system stops.

\[ \text{w.p. } p \]
\[ \text{w.p. } 1 - p \]

**Figure 13** Cyclic policy graph with implicit leaf node.

It is important to reiterate that the transition between nodes is random and exogenous to the rest of the problem (i.e. the states, controls, and noise). The two diagrams do not represent stopping problems where the agent can make the decision to exit the cycle. Instead, stopping problems can be modelled with a binary state variable representing whether the agent is stopped or not.

We are almost ready to describe the optimization problem of finding the optimal decision-rule for each node. However, before we proceed, it is necessary to introduce the concept of risk.

### 3.3 Risk

At every node in the policy graph of a multistage stochastic optimization problem, the agent needs some way of aggregating the future cost of a control across the future uncertainty. They do so via a risk measure.

**Definition** A risk measure $\mathcal{F}$ is a function that maps a random variable to a real number.

To elaborate upon this definition, we draw heavily from (Shapiro et al., 2009, Ch. 6.3). In this paper, we restrict our attention to random variables with a finite sample space $\Omega := \{ z_1, z_2, \ldots, z_K \}$ equipped with a sigma algebra of all subsets of $\Omega$ and respective (strictly positive) probabilities $\{ p_1, p_2, \ldots, p_K \}$. This greatly simplifies the analysis of risk and is a required assumption for our proposed solution technique (stochastic dual dynamic programming). We denote the random variable with the uppercase $Z$.

In this paper, we shall use the following risk measures: Expectation ($\mathbb{E}[Z]$), Average Value-at-Risk ($\text{AV@R}_{1-p}(Z)$), and Worst-case ($\max(Z)$). The Expectation and Worst-case risk measures are self-explanatory. However, the Average
Value-at-Risk is worth explaining for readers unfamiliar with it.

**Definition** According to Rockafellar and Uryasev (2002), the Average Value-at-Risk* at the $\beta$ quantile ($\text{AV@R}_{1-\beta}$) is:

$$
\text{AV@R}_{1-\beta}[Z] = \inf_{\xi} \left\{ \xi + \frac{1}{\beta} \sum_{k=1}^{K} p_k (z_k - \xi)_+ \right\},
$$

where $(x)_+ = \max(0, x)$.  

As a simple approximation, the $\text{AV@R}_{1-\beta}$ can be thought of as the expectation of the worst $\beta$ fraction of outcomes. However, if the distribution of the random variable is not continuous (e.g. the distribution is discrete), the interpretation is subtler since we may have to split a discrete probability atom. (See Rockafellar and Uryasev (2002) for more details.) Also note that when $\beta = 1$, $\text{AV@R}_{1-\beta}[Z] = \mathbb{E}[Z]$, and $\lim_{\beta \to 0} \text{AV@R}_{1-\beta}[Z] = \max[Z]$.

We use these risk measures (Expectation, AV@R, and Worst-case) because they are coherent according to the following axioms.

**Definition** A coherent risk measure is a risk measure $\mathbb{F}$ that satisfies the axioms of Artzner et al. (1999). For two discrete random variables $Z_1$ and $Z_2$, each with drawn from a sample space with $K$ elements, the axioms are:

- **Monotonicity**: If $Z_1 \leq Z_2$, then $\mathbb{F}[Z_1] \leq \mathbb{F}[Z_2]$.
- **Sub-additivity**: For $Z_1$, $Z_2$, then $\mathbb{F}[Z_1 + Z_2] \leq \mathbb{F}[Z_1] + \mathbb{F}[Z_2]$.
- **Positive homogeneity**: If $\lambda \geq 0$ then $\mathbb{F}[\lambda Z] = \lambda \mathbb{F}[Z]$.
- **Translation equivariance**: If $a \in \mathbb{R}$ then $\mathbb{F}[Z + a] = \mathbb{F}[Z] + a$.

We can also define coherent risk measures in terms of risk sets (Artzner et al., 1999; Shapiro et al., 2009). That is, a coherent risk measure $\mathbb{F}$ has a dual representation that can be viewed as taking the expectation of the random variable with respect to the worst probability distribution within some set $\mathbb{A}$ of possible distributions:

$$
\mathbb{F}[Z] = \sup_{\xi \in \mathbb{A}} \mathbb{E}_\xi [Z] = \sup_{\xi \in \mathbb{A}} \sum_{k=1}^{K} \xi_k z_k,
$$

(1)

where $\mathbb{A}$ is a convex subset of:

$$
\mathbb{A} = \left\{ \xi \in \mathbb{R}^K : \sum_{k=1}^{K} \xi_k = 1, \xi_k \geq 0 \right\}.
$$

Following Philpott et al. (2013), we shall refer to the probability distribution $\xi$ that attains the supremum of Eq. 1 as the changed probability distribution.

The three risk measures described above (Expectation, AV@R, and Worst-case) can be expressed in terms of the set $\mathbb{A}$ as follows:

- **Expectation**: If $\mathbb{A}$ is a singleton, containing only the original probability distribution, then the risk measure $\mathbb{F}$ is equivalent to the expectation operator.
- **AV@R**: If $\mathbb{A} = \left\{ \xi \in \mathbb{A} : \frac{\xi_k}{\xi_k} \leq \frac{\xi_k}{\xi_k}, \ k = 1, 2, \ldots, K \right\}$, then the risk measure $\mathbb{F}$ is equivalent to $\text{AV@R}_{1-\beta}$.

---

2 Rockafellar and Uryasev (2002) actually call this the Conditional Value-at-Risk (CV@R); however, we follow Shapiro et al. (2009) and refer to it as AV@R.
• **Worst-case:** If $\Psi = \Psi$, then $\mathbb{F}$ is the Worst-case risk measure.

### 3.4 Standard form

Let us recap what we have discussed so far. We can decompose a multistage stochastic optimization problem into a series of nodes. The nodes are linked together by state variables, and we can describe the linkages by a policy graph $G$. Associated with each node $i$ is a decision-rule $\pi_i(x_i, \omega_i)$ ($\pi_i(x_i)$ for Decision-Hazard), which maps the incoming state variable $x_i$ and realization of a random noise $\omega_i$, to a feasible control $u_i \in U_i(x_i, \omega_i)$ (in the Decision-Hazard case: $u_i \in U_i(x_i)$). As a result of taking the control $u_i$, the state transitions to the outgoing state $x'_i$ according to the transition function $x'_i = T_i(x_i, u_i, \omega_i)$, and a cost of $C_i(x_i, u_i, \omega_i)$ is incurred. Then, the system transitions to a new node in the policy graph according to the probability distribution of $\Phi$. All that remains is to define an optimization problem that can be used to find the optimal decision-rule $\pi_i$ for each node $i$. To do this, we utilize Bellman’s *principle of optimality* and form the cost-to-go for each node $i$, given the incoming state $x_i$ and realization of the noise $\omega_i$.

**Definition** *Principle of Optimality:* An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions. (Bellman, 1954)

**Definition** Given a policy $\pi$, the cost-to-go for node $i$ is the stage-objective as a result of taking the control $u_i$, plus the risk-adjusted cost-to-go of the node’s children:

$$V_i(x_i, \omega_i) = C_i(x_i, u_i, \omega_i) + \mathbb{E}_{j \in \Omega^i; \omega_j \in \Omega_j} [V_j(T_j(x_i, u_i, \omega_i), \omega_j)] ,$$

where $u_j = \pi_j(x_j, \omega_j)$ if the node is Hazard-Decision, and $u_i = \pi_i(x_i)$ if the node is Decision-Hazard.

Note that when we use the notation $\mathbb{E}_{j \in \Omega^i; \omega_j \in \Omega_j} [\cdot]$, we mean that we apply the risk measure to the distribution of outcomes after sampling all of the noise realizations in all of the children of node $i$. Since the noise realization and node transition are independent, from node $i$, the probability of sampling $\omega_j$ in the next node is $\phi_{i,j} \times p_{\omega_j}$ (i.e. the probability of transitioning from node $i$ to node $j$, multiplied by the probability of observing $\omega_j$ in node $j$). Also note that when maximizing, we refer to the value-to-go instead of the cost-to-go.

Putting together all that we have discussed so far, we can now define the class of multistage stochastic optimization problems that we consider in the remainder of this paper.

**Definition** Given a policy graph $G$, a **multistage stochastic optimization problem** is an optimization problem of the form:

$$\min_{\pi} \left\{ \mathbb{E}_{x_R \in \mathbb{R}^n; \omega_{\Omega_R} \in \Omega_R} [V_R(x_{\mathbb{R}}, \omega_{\Omega_R})] \right\} ,$$

(2)

where $x_R$ is the initial condition of the state variable at the root node.

Readers should note that this definition does not encompass all multistage stochastic optimization problems. For example, it does not consider continuous time problems, and due to the recursive definition of the cost-to-go function, we only use nested risk measures. However, this framework also allows single-stage risk measures within each subproblem (e.g. expected conditional risk measures (Homem-de-Mello and Pagnoncelli, 2016)), and end-of-horizon risk measures by dynamically changing the single-stage risk measures in each subproblem (e.g. Pflug and Pichler (2016); Baucke et al.
In addition, the non-anticipative constraints usually associated with multistage stochastic programming are satisfied by our definition of a policy.

The goal of multistage stochastic optimization is to find the policy $\pi$ that solves Eq. 2. If the policy $\pi$ is not given explicitly, the cost-to-go from the start of node $i$ can be formulated as an optimization problem. The formulation is different depending upon whether the node is Hazard-Decision or Decision-Hazard. We call the optimization problem associated with each node a subproblem.

**Definition** A Hazard-Decision subproblem is the optimization problem:

$$\text{HD}_i(x_i, \omega_i) : \quad V_i(x_i, \omega_i) = \min_{u_i} \quad C_i(x_i, u_i, \omega_i) + \mathbb{E}_{\omega_i \in \Omega_i} \left[ V_j\left(x_j', \omega_j\right) \right]$$

s.t. $x_i' = T_i(x_i, u_i, \omega_i)$

$u_i \in U_i(x_i, \omega_i)$

(3)

where the decision-rule $\pi_i(x_i, \omega_i)$ takes the value of $u_i$ in the optimal solution.

**Definition** A Decision-Hazard subproblem is the optimization problem:

$$\text{DH}_i(x_i) : \quad V_i(x_i) = \min_{\omega_i \in \Omega_i, j \in i^+} \mathbb{E}_{\omega_i \in \Omega_i} \left[ C_i(x_i, u_i, \omega_i) + V_j\left(x_j', \omega_j\right) \right]$$

s.t. $x_i' = T_i(x_i, u_i, \omega_i)$

$u_i \in U_i(x_i)$

(4)

where the decision-rule $\pi_i(x_i)$ takes the value of $u_i$ in the optimal solution.

Finally, a few last pieces of terminology. Given a policy $\pi$, we can simulate the multistage stochastic optimization problem by the procedure given in Algorithm 1. As a result, we end up with a sequence of nodes $i^n$, states $x^n$, controls $u^n$, noise terms $\omega^n$, and costs $c^n$ for $n \in \{1, \ldots, N\}$.

---

**Algorithm 1: Simulating the policy.**

1. Set $x^1 = x_R$
2. Set $n = 1$
3. While $i^{n-1} \neq \emptyset$
   1. Sample $i^n$ from $i^{n-1}$
   2. Sample $\omega^n$ from $\Omega_{i^n}$
   3. If $i^n$ is Hazard-Decision then
      1. Set $u^n = \pi_{i^n}(x^n, \omega^n)$
   4. Else
      1. Set $u^n = \pi_{i^n}(x^n)$
   5. Set $c^n = C_{i^n}(x^n, u^n, \omega^n)$
   6. Set $x^{n+1} = T_{i^n}(x^n, u^n, \omega^n)$
   7. Set $n = n + 1$

We also need to define a scenario and the cumulative cost of a scenario.
Definition A scenario is a sequence of node and noise realizations \((i^1, \omega^1), (i^2, \omega^2), \ldots, (i^N, \omega^N)\).

Definition The cumulative cost of a scenario is \(\sum_{n=1}^{N} c^n\).

3.5 Assumptions

So far in this paper, we have defined a general class of multistage stochastic optimization problems. In the next section, we describe an extension of the stochastic dual dynamic programming algorithm to this setting. However, our proposed solution technique requires a number of assumptions that restrict the class of problems the algorithm can solve. Given a policy graph \(G = (R, V, E, \Phi)\), these assumptions are as follows.

(A1) The number of nodes in \(V\) is finite.

(A2) Every node in the policy graph is hazard-decision.

(A3) The sample space \(\Omega_i\) of random noise outcomes is finite at each node \(i \in N\).

(A4) Given fixed \(\omega_i\) and excluding the risk-adjusted cost-to-go term, the subproblem associated with each node \(i \in N\) can be formulated as a convex optimization problem.

(A5) For every node \(i \in N\), there exists a bounded and feasible optimal control \(u_i\) for every reachable incoming state \(x_i\) and realization of the noise \(\omega_i\).

(A6) For every node \(i \in N\), the sub-graph rooted at node \(i\) has a positive probability of reaching a leaf node.

At first glance, these assumptions seem restrictive. However, let us make some comments regarding them. Assumptions (A1), (A2), (A3), and (A5) are standard assumptions in the stochastic dual dynamic programming literature Philpott and Guan (2008); Girardeau et al. (2015). Moreover, as we showed in Fig. 3, assumption (A2) is less taxing that first assumed as any decision-hazard node can be reformulated into two hazard-decision nodes with an expanded state-space. (This approach is used, e.g., by Street et al. (2018).) Assumption (A4) requires some other technical assumptions, such as an appropriate constraint qualification, but we will not detail them in this paper; see Girardeau et al. (2015) for details. Finally, assumption (A6) enforces a discounted infinite-horizon view of the world as opposed to an expected long-run average cost view of the world. It is also another way of saying that the discount factor around a cycle cannot be 1. (In the limit as the discount factor approaches 1, the discounted infinite horizon policy converges to the average cost policy (Bertsekas, 2005).)

To solve the multistage stochastic optimization problem formulated in this section, we propose a variant of the stochastic dual dynamic programming algorithm. There are many examples in the literature of similar algorithms and convergence proofs for various special cases of the policy graph. For example, in rough chronological order:

- Pereira and Pinto (1991) introduced the original stochastic dual dynamic programming (SDDP) algorithm for the case of a linear policy graph and a linear subproblem. However, the idea can be traced back to Benders decomposition (Benders, 1962) and the L-shaped method of Van Slyke and Wets (1969) (originally for two-stage problems, it was extended to the multistage case by Louveaux (1980) and Birge (1985)). SDDP has been studied extensively in the literature (Shapiro, 2011; Shapiro et al., 2013). We can also recommend the accessible introductions to SDDP provided by Newham (2008, Ch. 4), Guan (2008, Ch. 5), and Dowson (2018, Ch. 2).

- Many variants of the SDDP algorithm were presented in the literature, including AND (Donohue and Birge, 2006),
CUPPS (Chen and Powell, 1999), and ReSa (Hindsberger, 2014).
- Philpott and Guan (2008) proved the almost sure convergence of a modified version of SDDP. Earlier proofs, such as those by Chen and Powell (1999) and Linowsky and Philpott (2005), made use of an unstated assumption regarding the application of the second Borel-Cantelli lemma (Grimmett and Stirzaker, 1992). To differentiate their modified algorithm from the original SDDP, Philpott and Guan refer to their algorithm as DOASA.
- Various authors (Gjelsvik et al. (1999); Philpott and de Matos (2012)) presented algorithms for the Markovian policy graph setting, although none provided convergence results.
- Girardeau et al. (2015) proved the almost sure convergence of DOASA-type algorithms on a linear policy graph and a convex subproblem.
- Guigues (2016) proved the almost sure convergence of DOASA-type algorithms on problems with coherent risk measures.
- Rebennack (2016) presented an algorithm and convergence result for a general acyclic policy graph.
- Nannicini et al. (2017) presented an algorithm and convergence result for a linear policy graph that contains one cycle.
- Warrington et al. (2017) presented an algorithm and convergence result for a cyclic policy graph with a single node and no noise terms.
- Baucke (2018) presented a very similar result to Warrington et al. (2017), but used an exact upper bounding function to extend the analysis to a cyclic policy graph with any finite number of nodes, but no stagewise-independent noise terms. However, note that any node with stagewise-independent noise terms can be expanded into a collection of nodes with one node for each discrete realization of the noise term. Thus it is always possible to transform a problem from the setting used in this paper into the setting of Baucke.
- Guigues (2018) presented an algorithm and convergence result for problems with a random number of stages; these can be viewed as a linear policy graph with additional arcs between nodes such that the graph remains acyclic.

The algorithm we describe in the next section generalizes this body of work. The key difference in our work compared to the Markov decision process literature (e.g. Puterman (1994)) and the stochastic optimal control literature (e.g. Bertsekas (2005)) is that we consider continuous state and control variables.

4 | PROPOSED ALGORITHM

Due to our assumptions (A1)–(A6), the risk-adjusted cost-to-go function (i.e. \( F_{j \in I^1; \omega_j \in \Omega_j} \left[ V_j(x_j', \omega_j) \right] \)) is convex with respect to the state variable \( x_j' \). Therefore, it can be replaced by a variable \( \theta_j \) and approximated by the maximum of a set of affine functions, which we refer to as cuts. Like SDDP, our algorithm constructs the set of cuts iteratively. Each iteration consists of two phases: a forward pass, which samples a sequence of subproblems and values for the state variables \( S = [(i_1, x_{i_1}'), (i_2, x_{i_2}'), \ldots ] \); and a backward pass, which refines the approximation of the cost-to-go function by adding a new cut to each of the subproblems visited in the forward pass. The resulting approximated subproblem at
node $i$ after $K$ iterations can be expressed as:

$$
\text{SP}^K_i : \quad V^K_i(\bar{x}_i, \omega_i) = \min_{u_i, x'_i, \theta_i} C_i(x_i, u_i, \omega_i) + \theta_i \\
\text{s.t.} \quad x_i = \bar{x}_i, \quad [\lambda_i] \\
x'_i = T_i(x_i, u_i, \omega_i) \\
u_i \in U_i(x_i, \omega_i) \\
\theta_i \geq \alpha^k_i + \beta^k_i x'_i, \quad k \in \{1, 2, \ldots, K\}.
$$

(5)

Note that $\lambda_i$ is the vector of dual variables associated with the constraints $\bar{x}_i = x_i$. In other words, $\lambda_i$ is a valid subgradient for the function $V^K_i(\bar{x}_i, \omega_i)$ with respect to $\bar{x}_i$.

In the literature many authors (e.g., Pereira and Pinto (1991); Philpott and Guan (2008)) compute the subgradient of $V^K_i$ with respect to the incoming state variables $x_j$ by computing a transformation of the dual of the transition constraints (i.e. $x'_i = T_i(\bar{x}_i, u_i, \omega_i)$), taking into account the feasible set of actions $U_i$. This approach is overly complicated. Our solution (also used by Girardeau et al. (2015)) is to make $\bar{x}_i$ a dummy variable with the constraint:\n
$$\quad x_i = \bar{x}_i, \quad [\lambda_i].$$

This has downside of adding one extra variable and constraint for each state variable, but results in a simpler subgradient calculation.

### 4.1 Computing cut coefficients

We now explain how to calculate the cut coefficients $\alpha$ and $\beta$. Consider a node $j$, given an incoming state variable $\bar{x}$ and a realization of the noise $\omega_j$. We can solve $\text{SP}^K_j$ and record the optimal objective value $V^K_j(\bar{x}, \omega_j)$, which we denote $V^K_{j, \omega_j}$, and the optimal value of the dual variable $\lambda_j$, which we denote $\lambda^K_{j, \omega_j}$. Since $V^K_j$ is convex with respect to $x_j$, we have that:

$$
V^K_j(x_j, \omega_j) \geq V^K_{j, \omega_j} + \langle \lambda^K_{j, \omega_j}, (x_j - \bar{x}) \rangle,
$$

where $\langle, \rangle$ is the inner product. Re-arranging terms we get:

$$
V^K_j(x_j, \omega_j) \geq \left( V^K_{j, \omega_j} - \langle \lambda^K_{j, \omega_j}, \bar{x} \rangle \right) + \langle \lambda^K_{j, \omega_j}, x_j \rangle.
$$

However, recall that we wish to approximate the cost-to-go function

$$
\bar{F} = \mathbb{E}_{\omega_j} \left[ V_j(x_j', \omega_j) \right].
$$

To do so, consider the following proposition from (Philpott et al., 2013, Proposition 4), which we re-state using our notation without proof. (Recall that $\xi$ is the changed probability distribution from Section 3.3.)

**Proposition 1** Suppose, for each $\omega \in \Omega$, that $\lambda(\xi, x, \omega)$ is a subgradient of $V(\bar{x}, \omega)$ at $\bar{x}$. Then, given $\xi$ such that $\mathbb{E}[V(\bar{x}, \omega)] = \mathbb{E}_\xi[V(\bar{x}, \omega)]$, $\mathbb{E}_\xi[\lambda(\bar{x}, \omega)]$ is a subgradient of $\mathbb{E}[V(\bar{x}, \omega)]$ at $\bar{x}$.

---

3 Alexandre Street and Davi Valladão call this the fishing dual.
Using this result, we can construct a valid cut via the method given in Algorithm 2.

---

**Algorithm 2: Cut calculation algorithm.**

Solve $SP^K_i(x, \omega_j)$ for all $j \in i^+$ and $\omega_j \in \Omega_j$

Compute $\xi$ so that $\mathbb{E}_j \left[ \bar{V}_j^{K+1} \right] = \mathbb{E}_{j \in i^+} \mathbb{E}_{\omega_j \in \Omega_j} \left[ \bar{V}_j^{K+1} \right]$

Set $\beta_i^{K+1} = \mathbb{E}_j \left[ I_i^K \right]$ \ \ $\bar{V}_j^{K+1} = \left[ \bar{V}_j^{K+1} \right] - \langle \beta_i^{K+1}, \bar{x} \rangle$

Obtain the inequality $\mathbb{E}_{j \in i^+} \mathbb{E}_{\omega_j \in \Omega_j} \left[ V_j(x', \omega_j) \right] = \theta_i \geq a_i^{K+1} + \langle \beta_i^{K+1}, x'_i \rangle$

---

### 4.2 An iteration

We now describe an iteration of our algorithm in more detail. Each iteration consists of two phases: a forward pass, which samples a sequence of nodes and points in the state-space; and a backward pass, which uses the cut calculation algorithm to refine the approximation of the cost-to-go function at the points in the state-space visited on the forward pass.

In the forward pass of our algorithm, we sequentially solve a sequence of subproblems. After solving each subproblem, we terminate the forward pass and begin the backward pass with probability $1 - \sum_{j \in i^+} \phi_{ij}$. (Recall, $\phi_{ij}$ is the probability of transitioning from node $i$ to node $j$.) If we do not terminate the forward pass, we sample a new child node and continue the forward pass. Note also that if a node has no children (i.e. $i^+ = \emptyset$), then $\sum_{j \in i^+} \phi_{ij} = 0$ and we terminate the forward pass with probability 1. In addition, we choose a large finite $T$ and terminate the forward pass once we have sampled $T$ nodes in the sequence. We will discuss the choice of $T$ in Section 4.5. Pseudo-code for the forward pass is given in Algorithm 3.

---

**Algorithm 3: Forward pass.**

choose large, finite $T$

set $x = x_R$

set $S = [ ]$

set $i = R$

while $(rand() \geq 1 - \sum_{j \in i^+} \phi_{ij}) \land (|S| < T)$ do

\begin{itemize}
  \item sample new $i$ from $i^+$ according to the transition matrix $\Phi$
  \item sample $\omega_i$ from $\Omega_i$
  \item solve $SP^K_i(x, \omega_i)$
  \item append $(i, x'_i)$ to the list $S$
  \item set $x = x'_i$
\end{itemize}

end

---

Given a list of nodes visited on the forward pass (and corresponding sampled points in the state-space), the backward pass of our algorithm is identical to standard SDDP implementations. Pseudo-code for the backward pass is given in Algorithm 4. Note that because we may end the forward pass at a leaf node or due to a cycle, we need to check that a node has children before attempting to update the value function. This condition ($i^+ = \emptyset$) will only hold for the
last element in $S$ if the forward pass reached a leaf node. However, placing this check in the backward pass simplifies the logic in other parts of the algorithm. It is also important to note that one backward pass may add multiple cuts to the same node if the forward pass sampled that node multiple times before termination. Thus, there may be different numbers of cuts at different nodes in the graph. For notational simplicity, we shall continue to use $V_i^K$ to mean the approximated subproblem with $K$ cuts, and we assume that this always refers to the maximum number of cuts added to that subproblem.

**Algorithm 4: Backward pass.**

given $S$ from forward pass

for $(i, \hat{x})$ in reverse($S$) do

  if $i^+ = \emptyset$ then

    /* This check accounts for the case when the forward pass reached a leaf node. */

    continue

  end

  for $j \in i^+$ do

    for $\omega_j \in \Omega_j$ do

      solve $SP_j^K(\hat{x}, \omega_j)$

      set $\bar{\theta}_{j,\omega_j}$ to the optimal objective value

      set $\bar{\lambda}_{j,\omega_j}$ to the value of $\lambda_j$ in the optimal solution

    end

  end

compute $\xi$ so that

$$E_{\xi} \left[ \bar{\theta}_{j,\omega_j} \right] = E_{j \in i^+, \omega_j \in \Omega_j} \left[ \bar{\theta}_{j,\omega_j} \right]$$

set $\beta_{i+1}^K = E_{\xi} \left[ \bar{\lambda}_{j,\omega_j} \right]$

set $\alpha_{i+1}^K = E_{\xi} \left[ \bar{\theta}_{j,\omega_j} \right] - \langle \beta_{i+1}^K, \hat{x} \rangle$

add the cut $\theta_i \geq \alpha_{i+1}^K + \langle \beta_{i+1}^K, x_i' \rangle$ to $SP_i^K$

end

4.3 | Lower bound

Since the cuts at each node form an outer-approximation of the cost-to-go function, we can obtain a valid lower bound $y$ to the multistage stochastic optimization problem by evaluating:

$$y = \max_{i \in R^+; \omega_i \in \Omega_i} \left[ V_i^K(x_R, \omega_i) \right].$$

4.4 | Upper bound

We do not have an exact upper bound for this algorithm. This is not unique to our algorithm and is common in SDDP-type algorithms. (Some recent work has been done on upper bounds in the linear policy graph case, see e.g., Baucke et al. (2018); Leclère et al. (2018).) However, given all of the risk measures in the policy graph are the expectation operator, an unbiased estimate for the upper bound for the problem can be obtained by performing a Monte Carlo simulation
of the policy. This approach is commonly used in SDDP with acyclic policy graphs (see, e.g., Pereira and Pinto (1991)). Pseudo-code is given in Algorithm 5. In the risk-averse case, we do not have an upper bound.

**Algorithm 5: Upper bound calculation.**

choose large, finite $T$

for $n = 1, \ldots, N$

| set $x = x_R$
| set $i = R$
| set $y_n = 0$
| while ($\text{rand}() \geq 1 - \sum_{j \in i^+} \phi_{ij}) \land (|S| < T)$ do
| sample new $i$ from $i^+$ according to the transition matrix $\Phi$
| sample $\omega_i$ from $\Omega_i$
| solve $SP^K_j(x, \omega_i)$
| set $y_n = y_n + V^K_j(x, \omega_i)$
| end

end

Calculate the sample mean

$\bar{y} = \frac{1}{N} \sum_{n=1}^{N} y_n$

Form a confidence interval around $\bar{y}$ for the population mean based on $\{y_n\}_{1}^{N}$

### 4.5 Convergence

In this section, we prove the finite $\epsilon$–convergence of our algorithm for a simple cyclic policy graph containing one node that loops back onto itself (Fig. 14). Despite the simple setting, readers should easily see how the proof can be extended to more complicated policy graphs.

![Cyclic policy graph with implicit leaf node.](image)

**FIGURE 14** Cyclic policy graph with implicit leaf node.

By assumption (AS), for every node $i \in \mathcal{N}$, there exists a bounded and feasible optimal control $u_i$ for every reachable incoming state $x_i$ and realization of the noise $\omega_i$. Thus, there exists some large positive bound $M$ on the stage-objective of all nodes such that for all nodes $i$, reachable incoming states $x_i$, and realizations of the noise $\omega_i$:

$$ -M \leq C_i(x_i, u_i^*, \omega_i) \leq M, $$
where $u^*_j$ is the optimal control. Thus, for the policy graph in Fig. 14, a valid bound on $\mathbb{E}_{j \in \mathbb{R}^+; \omega_j \in \Omega_j} V(x'_j, \omega_j)$ is

$$\frac{-M}{1-\rho} \leq \mathbb{E}_{j \in \mathbb{R}^+; \omega_j \in \Omega_j} \left[ V(x'_j, \omega_j) \right] \leq \frac{M}{1-\rho}. \tag{6}$$

By assumption (A6), $\rho < 1$, and so $M/(1 - \rho)$ is finite. Now consider unrolling the cycle of the policy graph in Fig. 14 so that it contains $T$ nodes. We refer to $T$ as the maximum depth. An example where $T = 4$ is given in Fig. 15.

**Figure 15** Unrolled cyclic policy graph with implicit leaf nodes in the case where $T = 4$.

Based on the upper bound derived in Eq. 6, we know that in the last node, a bound on the cost-to-go that we have not modelled (because we truncated the infinite horizon problem to $T$ nodes) is:

$$\begin{bmatrix} \rho^T \times \frac{-M}{1-\rho}, & \rho^T \times \frac{M}{1-\rho} \end{bmatrix}. \tag{7}$$

Given some large, finite $T$, the unrolled policy graph is no longer cyclic. Instead, it is an acyclic Markovian policy graph. In this setting, our algorithm is identical to other SDDP-type algorithms that have proven finite convergence (see, e.g., Philpott and Guan (2008); Girardeau et al. (2015); Guiges (2016)). The probabilistic termination of the forward pass is equivalent to reaching a dummy leaf node with $C_i(x_i, u_i, \omega_i) = 0$. However, because of the truncation of the number of nodes, we do not obtain the optimal policy. Instead, we obtain an $\epsilon$-optimal policy such that

$$\mathbb{E}_{j \in \mathbb{R}^+; \omega_j \in \Omega_j} \left[ V(x_R, \omega_j) \right] - \mathbb{E}_{j \in \mathbb{R}^+; \omega_j \in \Omega_j} \left[ \tilde{V}^*(x_R, \omega_j) \right] \leq \epsilon \leq \rho^T \times \frac{|M|}{1-\rho}, \tag{8}$$

where $\tilde{V}^*$ is the converged cost-to-go function of the truncated problem. This error $\epsilon$ can be made arbitrarily small by choosing a large, finite $T$.

Although the argument presented above is for the simplest cyclic policy graph, a similar argument can be constructed for arbitrarily complex cyclic policy graphs. The difference is that it was trivial for us to derive a bound on $\epsilon$ as a function of the maximum depth $T$. In general, the relationship between $T$ and $\epsilon$ is dependent upon the specific policy graph. However, because of assumption (A6), the bound on $\epsilon$ is finite. Thus, given a bound on $\epsilon$, any cyclic policy graph can be unrolled to form an acyclic policy graph with a maximum depth of $T$ nodes and solved to an $\epsilon$-optimal solution.

We note that our algorithm, and this convergence argument, is very similar to the work of Nannicini et al. (2017). The key difference is that we choose a maximum depth $T$ apriori and use the transition probabilities to terminate the forward pass, whereas Nannicini et al. (2017) use an iterative method to slowly increment the value of $T$ until the required $\epsilon$-convergence is achieved and do not use the transition probabilities to terminate the forward pass. Moreover, due to policy graph formulation, our algorithm applies to arbitrary graphs, as opposed to the linear policy graph with a single cycle considered by Nannicini et al. (2017). However, it is likely that their method also extends to the arbitrary policy graph case.

We also highlight the work of Legat and Jungers (2016) who (in an application arising from information theory and not stochastic programming) proposed an algorithm for a cyclic policy graph when $C_i(x_i, u_i, \omega_i) = 0$. Like our algorithm,
their algorithm includes a maximum depth limit that is used to terminate the forward pass. Notably, their work considers problems without relatively complete recourse, and so Benders feasibility cuts are generated at each node in the graph.

5 | EXAMPLE: PASTORAL DAIRY FARMING

To demonstrate the practicality of our algorithm, we solve a single instance of a multistage stochastic optimization model from the literature. Since we only solve this one instance, we will not draw conclusions about the computational performance of the method, or the impact on solution times and policy quality of choosing different parameters (such as the discount factor). We leave these tasks for future work.

The POWDER model of Dowson et al. (To appear) is a multistage stochastic optimization model of a pastoral dairy farm over the course of one year. POWDER decomposes the year into a Markovian policy graph with 52 weekly stages. The decision process is modelled as follows. At the beginning of week $t = 1, 2, \ldots, 52$, the farmer measures the five state variables in the model: the soil moisture $W_t$ (mm), the pasture cover $P_t$ (kg/ha), the quantity of grass in storage $Q_t$ (kg/ha), the number of cows milking $C_t$ (cows/ha), and the quantity of milk solids produced to date $M_t$ (kg/ha). Before choosing an action for the week, the farmer observes the realization of the two stagewise-independent random variables: the potential evapotranspiration $e^p_t$ (mm), and the quantity of rainfall $r_t$ (mm). The Markov states in the Markovian policy graph correspond to a forecast for the price that the farmer receives for their milk in the last stage of each year ($$/kg). We refer to this price as the end-of-season milk price. During the year, we refer to the forecast for the end-of-season milk price associated each Markov state as the forecast milk price. Taking into account the incoming values of the state variables, the Markov state $p_t$, and the observation of the stagewise-independent noise term, the farmer decides the quantity of pasture to harvest $h_t$ (kg/ha), the number of cows to stop milking $u_t$ (cows/ha), and the quantities of grass from pasture $f_t^p$, grass from storage $f_t^q$, and palm kernel $s_t$ to feed the herd (all kg/ha). As a result of these actions, the system transitions to a new state that will serve as the incoming state in the next week, and the farmer incurs the cost of purchasing palm kernel, harvesting pasture, and applying irrigation. In the last week, the forecast milk price becomes the end-of-season milk price, and the farmer sells all of the milk produced during the season $M_{52}$ (kg/ha) at the end-of-season milk price $p_{52}$ ($$/kg).

As a simple example of the dynamics in the model, consider $Q_t$ (kg/ha), the quantity of grass in storage at the start of week $t$. This is a state variable with linear dynamics:

$$Q_{t+1} = Q_t + \beta h_t - f_t^q,$$

where $h_t$ (kg/ha) is the quantity harvested, $\beta$ is the harvesting efficiency $< 1$, and $f_t^q$ (kg/ha) is the amount of grass from storage fed to the cows. As a result, the farmer incurs a cost of $c^h \times h_t$ in week $t$, where $c^h$ is the cost of harvesting one kg of grass (a constant in the model).

POWDER is able to model the effect of weather and price uncertainty on the management actions of the pastoral dairy farmer. However, because of the finite-horizon assumption, the authors introduced a penalty if the pasture cover at the end of the year was less than the pasture cover at the start of the season. In this section, we relax this constraint and model POWDER as an infinite horizon problem. We retain the 52 weekly stages and Markovian policy graph structure used in Dowson et al. (To appear), and assume that the end of week 52 is equivalent to the start of week 1. A graphical representation of the policy graph is given in Fig. 16. We assume that the discount factor associated with the arcs exiting nodes in the 52nd stage is $< 1$.

---

4 A function of sunlight and temperature that is positively correlated to grass growth.
5.1 | Results

Based on our domain knowledge of the POWDER model, we can derive an lower bound for the expected value of each season by assuming that no milk is produced and all of the energy required by the cows is derived from palm kernel. Using the data from Dowson et al. (To appear), a conservative value for this lower bound is $–$11,000/ha/year. In addition, we can derive an upper bound for the expected value of each season by assuming that every cow produces the maximum quantity of milk, that the farmer sells this milk at the expected end-of-season milk price, and that they incur no costs. Using the data from Dowson et al. (To appear), a conservative value for this upper bound is +$10,000/ha/year. Thus, we can obtain a bound on $\epsilon$ as a function of the discount factor $\rho$ and maximum depth $T$ of:

$$
\epsilon \leq \rho^{T/52} \times \frac{11,000}{1 - \rho}.
$$

We solved the infinite horizon POWDER model for 500 iterations using a discount factor of $\rho = 0.75$ and maximum depth of $T = 780$. Thus, from Eq. 9, we know that $\epsilon \leq$ $587$/ha. After iterations 100, 200, 300, 400, and 500, we performed a Monte Carlo simulation of the policy with 1000 replications to obtain a confidence interval for the lower bound. (Note that upper and lower are reversed since POWDER is a maximization model.) A plot of the lower and upper bounds as a function of the number of iterations is given in Fig. 17.

We also simulated the policy after 500 iterations with a Monte Carlo simulation of 1000 replications. Unlike the simulation used to obtain an estimate for the lower bound, we did not terminate the forward pass with probability $1 - \rho$ at the end of each season. Instead, we terminated the forward pass once five complete seasons had been simulated. This simulation is visualized in Fig. 18. In each of the subplots, we plot the 0–100 percentiles of the distribution of the plotted variable as a light shaded band. The dark shaded bands correspond to the 10–90th percentiles. The dotted line corresponds to the 50th percentile. Fig. 18a shows the number of cows that are milking (a state variable) at the end of each stage. We start each season (August 1st) will all cows milking (i.e. 3 cows/ha). At some point during the season, the farmer stops milking the cows, and the number of cows milking drops to zero. In Figure 18b we plot the pasture cover (a state variable) at the end of stage. Excluding the first season, the pasture cover exhibits strong seasonality with a minimum around 500 kg/ha in July and a maximum of 2000 kg/ha in December. This seasonality is driven by the
**FIGURE 17** Convergence of the upper bound against the number of iterations for the POWDER model with a discount factor of $\rho = 0.75$. Thick vertical bars are 95% confidence intervals (constructed using the modified Cox approach of Olsson (2005)) for the mean of the Monte Carlo simulation of the policy.

evapotranspiration (a stagewise-independent noise), which we plot in Figure 18c. Since evapotranspiration is a function of the incoming solar radiation and air temperature, it has a maximum in the New Zealand summer (Dec - Feb) and a minimum in the New Zealand winter (Jun - Aug).

**FIGURE 18** Simulation of the infinite horizon policy over five seasons.

In Dowson et al. (To appear), the authors placed a large penalty on ending the season with a pasture cover of less than 2,500 kg/ha. They justified this by arguing that it was necessary for the pasture cover at the end of the year to be no less than the pasture cover at the beginning of the year in order for the same policy to be used in consecutive years. However, as shown in Figure 18b, this constraint is sub-optimal. Instead of ending the year with a pasture cover of 2,500 kg/ha (above the long-run maximum), the optimal solution in the infinite horizon policy is to end the season with the minimum pasture cover, enabling the pasture cover to synchronize with the evapotranspiration. We can also compare the discounted expected value the finite horizon and infinite horizon policies. In the finite horizon case, POWDER converges to a value of $5,782/ha/year. Discounting this value at a discount factor of $\rho = 0.75$ gives a discounted expected value of $23,128/ha. In comparison, the infinite horizon policy converges to a value of $25,035/ha ± $587/ha (accounting for the $\epsilon$ error). Thus, the infinite horizon policy leads to an increase in value of at least $1,320/ha (+5.7\%) in the discounted expected value. This demonstrates the value that can be gained by solving the infinite horizon problem as opposed to a finite horizon problem in which the user has to provide the terminal value function.
6 CONCLUSION

This paper introduced the policy graph as a structured way of formulating multistage stochastic optimization problems. The policy graph representation unifies many of the existing special case formulations of multistage stochastic optimization problems in the literature into one general framework. This paper also proposed an extension to the stochastic dual dynamic programming algorithm to the case of a general policy graph. By truncating the infinite horizon problem with some small bounded error \( \epsilon \), we are able to recover an acyclic policy graph for which there are existing proofs of convergence.

Although this paper has presented one application from agriculture, there are many more applications that this could be applied to. Indeed, in many real-world applications, the natural model is an infinite horizon formulation as opposed to a finite horizon formulation.

SUPPLEMENTARY MATERIALS

To facilitate the adoption of our algorithm, we have developed Kökako, an open-source package in the Julia language (Bezanson et al., 2017). Source code, tutorials, and documentation can be found at github.com/odow/Koko. jl.

Acknowledgements

We thank David Morton, Tony Downward, Andy Philpott, Andrew Mason, Vincent Leclère, François Pacaud, Benoît Legat, and Joaquim Dias Garcia for the many useful discussions that directly led to this work.

REFERENCES


