SOLIB: A Benchmark Library for Stochastic Optimization

Oscar Dowson[a,*]

[a]*Department of Engineering Science at The University of Auckland, 70 Symonds Street, Auckland 1010, New Zealand*

Abstract

The field of computational stochastic optimization is highly fragmented. In this paper we propose a graphical framework for communicating a large class of computational stochastic optimization models, and define a common file-format for representing such problems. We have collated the beginnings of a test-set of real-world models (named SOLIB - Stochastic Optimization Library) and used SOLIB to perform a comparison between to variations of the Stochastic Dual Dynamic Programming algorithm.

Keywords: multistage, computational stochastic optimization, stochastic programming

1. Introduction

In 1974, Klingman et al. [1] wrote "one of the problems ... in trying to benchmark codes based on different methodologies ... [is] their lack of uniformity for input specification. This nonstandardization of problem specification ... is most frustrating and has hampered benchmarking since researchers are reluctant to recode their input routines." In 2007, Gassmann and Kristjansson [2] highlighted this remark and compared it to the then current state of stochastic optimization. However, as far back as 1987, Birge et al. [3] wrote that efforts to solve stochastic optimization problems "have tended to be directed towards specialized types of problems resulting in a pot-pourri of numerical codes, problem formulations and data sets that are not comparable or even mutually compatible. The development of general purpose codes suitable for industrial application is hampered by the lack of a comprehensive set of test problems in a standard format to compare performance and challenge competitors." The file format they laid out in [3], now called SMPS (Stochastic Mathematical Programming System), was a format they "hope[d] will become the nucleus of a standard problem format for multistage stochastic linear programs."

Despite this hope, twenty years later, in 2007, Gassmann and Kristjansson wrote [2] that "while [the SMPS format] is widely used, SMPS does not enjoy universal acceptance. Part of the reason is that the record-based structure of MPS is deemed to be overly rigid and limiting, but we feel that at least in part the reason is a lack of examples that describe the many and varied constructs of the SMPS format."

Now, ten years later still, and thirty years after the original paper by Birge et al., a number of test-sets have been collated [4, 5, 6, 7, 8]. While these appear to demonstrate the flexibility of

*Corresonding author

Email address: odow003@aucklanduni.ac.nz (Oscar Dowson)

the SMPS standard, we highlight a number of features that are, in our view, large short-comings. Namely, these are

1. The diverging scenario-tree based approach. It is impossible[1] for a child node to be descended from two parents. This is important in order to avoid the exponential growth in the number of leaves of the scenario tree;
2. The reliance on the MPS standard (neither human readable, nor machine efficient);
3. The reliance on temporal ordering of coefficients in the Core file;
4. The lack of a clear definition of a state variable;
5. The inability to specify non-quadratic terms.

Furthermore, despite the large number of test-problems, tools for reading, writing, and solving problems in the SMPS standard are not widespread. We hypothesize that the reason for this is not deficiencies in the SMPS standard, but an overall fragmentation of the community in general.

In recent years, Powell [9, 10] has highlighted the fragmentation of our field, and made the case for unifying the various communities of Stochastic Optimization (which goes by various names including Optimal Control, Stochastic Search, Dynamic Programming, Robust Optimization, etc.) under the umbrella of *Computational Stochastic Optimization* (henceforth referred to as CSO). Indeed, for terms as widely used as *state* and *stage*, there seems to be no widely agreed upon definition.[2] This makes it hard to develop standards and share work when we cannot even agree what it is that we are trying to share.

Another reason for the limited uptake may be the lack of an open-source, unified set of modelling languages and solution algorithms in modern computational languages (e.g. Julia, Python). Consider the problem of solving a dynamic program (not even necessarily stochastic) via backwards recursion. The practitioner must re-invent the wheel, and usually codes such an implementation in a low-level language such as C or C++ to get performance. They usually seek the path of least resistance and implement the bare-minimum required to solve a particular problem or prove a particular algorithm's performance. They are unlikely to go to the effort of implementing an SMPS reader (let alone writer), and therefore find no incentive to copy their new models across to a standard they don't support.

SMPS is not the only file-format that has been proposed to stochastic optimization problems. Another competing file-format that can be used to represent stochastic optimization problems is the OSiL (Optimization Services instance Language) file-format [11]. OSiL uses an XML based schema for representing problem data. This means the file is more human readable (and editable) than the SMPS files. In addition, it is able to specify arbitrary non-linear terms by creating a nested expression graph. Unlike the SMPS format, the OSiL format has a number of different ways of specifying the uncertainty. These include explicit scenario trees (like the SMPS format), as well as implicit scenario trees by describing the different random variables at each stage. However, OSiL makes the same error as the SMPS format, in that, despite decomposing the problem into *stages*, and identifying random variables and non-anticipativity constraints, it fails to clearly identify state variables in the canonical form it can support. In our view, this leads to a format that is more complicated than necessary.

We echo the words of Birge et al., and believe that the CSO community has been held back by this fragmentation. Therefore, we propose four goals which should be given top priority by the community:

[1] I haven't found an example or documentation to say otherwise.

[2] See Powell's comment in [9] "there is an astonishing lack of agreement in the definition of a state variable."

1. The standardization of definitions and problem specifications from a mathematical standpoint;
2. The creation, and adoption, of a common file-format for sharing CSO problems;
3. The maintenance of an archive of CSO problems in that file-format;
4. The development of a suite of inter-operable, open-source modeling libraries and solvers for CSO.

In this paper we attack the first three points. We refer the reader to (TODO `SDDP.jl` paper) for a discussion of some recent advancements in respect to the fourth point. In section 2, we discuss the staging of computational stochastic optimization problems (namely the question: what is a stage?) and present the idea of a *stage graph*. In section 3, we outline the canonical form we use to represent CSO problems that arises naturally from the concept of a stage graph. Lastly, in section 4, we propose a new file-format for a sub-class of CSO problems (those with discrete random variables with known probabilities) that is a close representation of the canonical form from section 3. This new file-format overcomes many of the limitations identified in the SMPS and OSiL file-formats, and clearly identifies the decomposition structure of the problem.

2. Staging of CSO problems

In [9], Powell makes the case that almost all stochastic optimization problems can be formulated as a sequential decision problem with an exogenous, memoryless (markovian), information process. For this paper we ignore continuous time problems and only consider problems where the sequential decision making happens in discrete time. Continuous time problems have been extensively studied and can be more easily described by partial differential equations, rather than the approach we advocate in this paper.

Powell identifies the five dimensions of any stochastic optimization problem as:

1. State variables: We take the definition from [10] as:
   The state variable is the minimally dimensioned function of history that captures all the information we need to model a system from some point in time onward.
   These state variables can be continuous or discrete. We denote them with the lowercase x_t at time t.
2. Control variables: A control variable is an action or decision taken by the operator (whether explicitly or implicitly) that changes the state of the system.
   These can be continuous or discrete. We denote them with the lowercase u_t at time t.
3. Exogenous information (the uncertainty). A noise is a (potentially multi-dimensional) random variable that is exogenous to the state of the system and the controls chosen by the operator. We denote a single observation of the random variable with the lowercase ω_t and the distribution from which it is drawn by the uppercase Ω_t at time t.
4. Transition function: the transition function is a (potentially unknown) mapping of the current state x_t, the control u_t and a realization of the noise ω_t to a new state x'_t.
5. Objective function: the cost (reward) accrued by taking action u_t in state x_t and realization ω_t.

We believe this list is missing one important item: an explicit representation of the sequential ordering of the decision making (in other words, how the exogenous information is revealed over the decision making process). In theory, this information is implicit within the five dimensions, however as we shall see, there are clear benefits from making the staging process explicit.

The core decomposition feature of a computational stochastic optimization problem is the stage. There are two types of stages:

1. Hazard-Decision. In a Hazard-Decision stage (Figure 1), the operator chooses an action u_i after observing a realization of the random variable $\omega_i \sim \Omega_i$ according to a policy $\pi_i(x_i, \omega_i)$. The state transitions from x_i to x_{i+1} according to the transition function $T_i(x_i, u_i, \omega_i)$. The policy respects the set of admissible actions so that $u_i \in U_i(x_i, \omega_i)$ and $x_{i+1} \in \mathcal{X}_i$. In addition, a cost $C_i(x_i, u_i, \omega_i)$ is incurred.

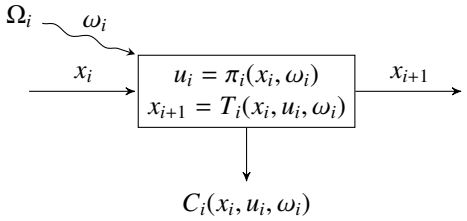

Figure 1: Hazard-Decision stage.

2. Decision-Hazard. In a Decision-Hazard stage (Figure 2), the operator chooses an action u_i before observing a realization of the random variable $\omega_i \sim \Omega_i$ according to a policy $\pi_i(x_i)$. The state transitions from x_i to x_{i+1} according to the transition function $T_i(x_i, u_i, \omega_i)$. The policy respects the set of admissible actions so that $u_i \in U_i(x_i)$ and $x_{i+1} \in \mathcal{X}_i$. In addition, a cost $C_i(x_i, u_i, \omega_i)$ is incurred.

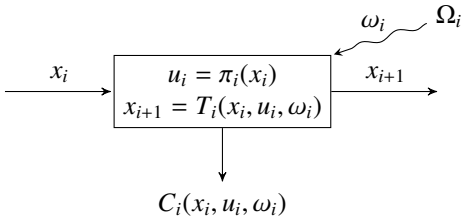

Figure 2: Decision-Hazard stage.

When there is no uncertainty that is realized in the stage, the Decision-Hazard stage becomes identical to the Hazard-Decision stage. We denote this deterministic stage by dropping the wavy line coming into the stage.

2.1. Stage Graph

CSO problems usually involve more than one stage chained in sequence. In this section we show by example a graph based approach to defining multi-stage problems that is able to describe any combination of sequential decision making processes. Shaded gray circles are the root nodes. A root node contains the initial value of the state variables, and is typically the point in the decision making process that we calculate the objective of the optimization problem.

We call the graph describing the stages the *stage graph*. For simplicity, we drop the textual annotations used above as their values can be inferred based on the type of symbol used.

Deterministic Sequential Decision problem. In the deterministic setting, the system is in a given state, and action is then chosen that transitions the system to a new state (Figure 3).

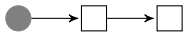

Figure 3: Two-stage Decision-Only problem.

Sequential Hazard-Decision problem. In a sequential Hazard-Decision problem (Figure 4), the noise is observed before the action is taken.

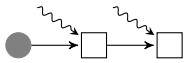

Figure 4: Two-stage Hazard-Decision problem.

A stage graph can also be thought of as a compressed scenario tree. It is also possible to express the problem as an explicit scenario tree. If each Hazard-Decision stage in Figure 4 has two possible realizations for ω, the scenario tree has six Decision-Only stages (Figure 5). The uncertainty lies on the exiting arcs from the stage problems.

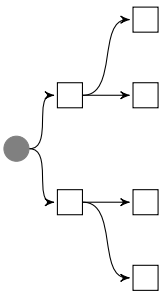

Figure 5: Explicit scenario-tree representation.

Any stage graph (with finite, discrete, distributions for Ω) can be expanded into a scenario tree, but a scenario tree cannot be compressed into a stage graph without information about which branches are independent.

Sequential Decision-Hazard problem. In a sequential Decision-Hazard problem (Figure 6), the noise is observed after the action is chosen.

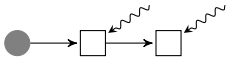

Figure 6: Staged Here-and-Now (Decision-Hazard) Problem.

Two-stage with recourse. A common problem in the literature is the two-stage problem with recourse (Figure 7). In this problem, the operator chooses a first stage decision u_1, realizes some uncertainty, and then can take a recourse decision based on the observation of the uncertainty.

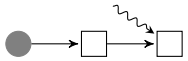

Figure 7: Two-stage recourse problem.

2.1.1. Conditional-Dependence

In this section we depart slightly from Powell's claim that all properly modeled systems are markovian. It can be the case that when a state variable depends *only* on the exogenous information process (and not on the rest of the state variables, or the control variables), then it is possible to combine both a markovian information process and a discrete scenario tree.

For a motivating example for Figure 8, consider the following: The El Niño-Southern Oscillation is a 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 the Western Pacific. In La Niña years, the opposite is true. When the first decision is taken, it is unknown whether the year will be El Niño or La Niña. Once this is revealed, a recourse decision can be taken, before the actual rainfall is observed.

We could model this as a single branch, two-stage problem like Figure 7, but we would need to encode an additional state variable (the belief about whether the system was in El Niño or La Niña).

To model conditional-dependence in the information process, we allow multiple arcs to exit from a stage. Each arc is associated with a probability such that the sum of the probabilities on the arcs leaving a stage sums to one. In addition, multiple arcs can enter a stage.

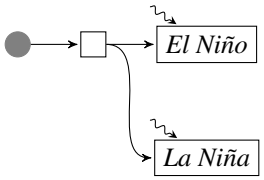

Figure 8: Two-stage recourse problem with conditional dependence.

It is also possible for the conditional-dependence to be markovian (Figure 9). This recombination of the scenario tree helps prevent the number of leaf nodes from growing exponentially.

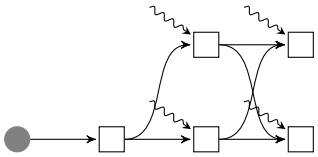

Figure 9: Multi-stage recourse problem with markovian conditional dependence.

Another reason for introducing a scenario tree is an example like Figure 10. In one branch of the tree, there are two sequential decisions, in the other, there are three. One way to consider this problem is that there are three sequential decisions in all branches, and that the feasible action space of the third decision on the lower branch is the null set. Furthermore, some stages are deterministic, some are Decision-Hazard, and some are Hazard-Decision.

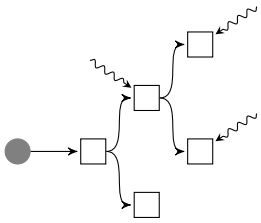

Figure 10: Multi-stage recourse problem conditional dependence.

2.1.2. Cyclic Graphs

It is also possible to capture Infinite Horizon problems with the graph based approach (Figure 11. These can be feedback loops (for example, from stochastic optimal control), or even sub-graphs embedded inside a larger graph (Figure 12).

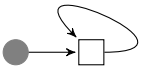

Figure 11: Infinite Horizon Problem.

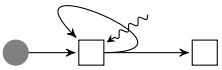

Figure 12: Infinite Horizon Problem.

Typically, these involve the need to add a *discount factor* in order to discount the future costs. The question arises how to incorporate these into the *stage graph*. The solution we propose, it to allow the probabilities of the arcs exiting a bellman node to sum to less than one. Implicitly, this results in the addition of an additional leaf stage with zero cost (Figure 13). The *discount factor* λ can be interpreted as a 1 - probability that the system stops.

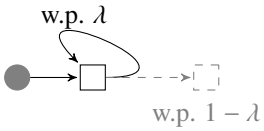

Figure 13: Cyclic graph with explicit discount.

3. Canonical form of a CSO Problem

In [10], Powell proposed the canonical form for a CSO problem as

$$\max_{\pi} \mathbb{E}\left\{\sum_{t=0}^{T} C(S_t, X_t^{\pi}(S_t)) \mid S_0\right\}, \tag{1}$$

where decisions are made according to a policy $x_t = X_t^{\pi}(S_t)$, where we might use $a_t = A_t^{\pi}(S_t)$ for discrete actions, or $u_t = U_t^{\pi}(x_t)$ for state x_t and control u_t. States evolve according to a (possibly unknown) transition function $S_{t+1} = S^M(S_t, X_t^{\pi}(S_t), W_{t+1})$.

This desire for a compact canonical form is admirable (it is hard to beat $min\{c^\top x \mid Ax = b\}$), but it unnecessarily obscures details about the staging of a CSO problem (it is also unclear how to deal with nested risk-measures). We prefer the recursive definition of (1) for CSO problems:

$$\min_{\pi} \mathop{\mathbb{F}}_{i\in 0^+;\ \omega\in\Omega_i}\left[V_i^{\pi}(x_0,\omega)\right] \mid x_0, \tag{2}$$

where

$$V_i^{\pi}(x_i,\omega) = C_i(x_i,u_{i,\omega},\omega) + \mathop{\mathbb{F}}_{j\in i^+,\ \omega_j\in\Omega_j}\left[V_j^{\pi}(x_{i,\omega},\omega_j)\right], \tag{3}$$

$x_{i,\omega} = T_i(x_i, u_{i,\omega}, \omega)$. If the stage i is Hazard-Decision $u_{i,\omega} = \pi_i(x_i,\omega)$, otherwise the stage is Decision-Hazard and $u_{i,\omega} = \pi_i(x_i)$. The policy respects any constraints on the domain of the control so that $u_{i,\omega} \in U_i(x_i,\omega)$ if the stage is Hazard-Decision and $u_{i,\omega} \in U_i(x_i)$ if the stage is Decision-Hazard. $\mathbb{F}$ is a risk measure. i^+ is the set of child nodes of stage i. There is a known, finite probability of transitioning from stage i to every stage in i^+. The set of nodes 0^+ are the children of the root node. We also assume relatively complete recourse of x_i. That is, for all $\omega \in \Omega_i$ and possible incoming values of x_i, there is a feasible control $u_{i,\omega}$, with a bounded objective value $V_i^{\pi}(x_i,\omega)$.

In the stochastic optimization community, Hazard-Decision policies are typically constructed by forming a mathematical optimization problem. Therefore, $V_i^{\pi}(x_i,\omega)$ is the objective of the optimization problem:

$$\begin{aligned}
\mathbf{HD}_i(x_i,\omega): \quad \min_{u_{i,\omega}} \quad & C_i(x_i,u_{i,\omega},\omega) + \mathop{\mathbb{F}}_{j\in i^+;\ \varphi\in\Omega_j}\left[V_j^{\pi}(x_{i,\omega},\varphi)\right] \\
s.t. \quad & x_{i,\omega} = T_i(x_i,u_{i,\omega},\omega) \\
& u_{i,\omega} \in U_i(x_i,\omega),
\end{aligned} \tag{4}$$

and the policy $\pi_i(x_i,\omega) \in \arg\min_{u_{i,\omega}} \mathbf{HD}_i(x_i,\omega)$. The future cost component of the objective is usually non-trivial, and so approximation schemes are used.

A Decision-Hazard policy can also be formed as the optimization problem:

$$\begin{aligned}
\mathbf{DH}_i(x_i): \quad \min_{u_i} \quad & \mathop{\mathbb{F}}_{\omega\in\Omega_i}\left[C_i(x_i,u_{i,\omega},\omega) + \mathop{\mathbb{F}}_{j\in i^+;\ \varphi\in\Omega_j}\left[V_j^{\pi}(x_{i,\omega},\varphi)\right]\right] \\
s.t. \quad & x_{i,\omega} = T_i(x_i,u_{i,\omega},\omega) \\
& u_i \in U_i(x_i) \\
& \underbrace{u_i = u_{i,\omega_1} = \cdots = u_{i,\omega_N}}_{\text{Non-anticipativitiy constraint}},
\end{aligned} \tag{5}$$

where the policy $\pi_i(x_i) \in \arg\min_{u_i} \mathbf{DH}_i(x_i)$, and $V_i^{\pi}(x_i,\omega) = C_i(x_i,u_{i,\omega},\omega) + \mathop{\mathbb{F}}_{j\in i^+;\ \varphi\in\Omega_j}\left[V_j^{\pi}(x_{i,\omega},\varphi)\right]$.

Equation (5) can also be expressed as the two-stage stochastic program:

$$\begin{aligned} \mathbf{DH}_i(x_i): \quad & \min_{u_i} \quad \mathop{\mathbb{F}}_{\omega\in\Omega_i} \left[Q_i(x_i, u_i, \omega)\right] \\ & s.t. \quad u_i \in U_i(x_i), \end{aligned} \tag{6}$$

where $Q_i(x_i, u_i, \omega)$ is the optimal second-stage objective value of

$$\begin{aligned} Q_i(x_i, u_i, \omega) = \quad & \min_{x_{i,\omega}} \quad C_i(x_i, u_i, \omega) + \mathop{\mathbb{F}}_{j\in i^+;\ \varphi\in\Omega_j} \left[V_j^{\pi}(x_{i,\omega}, \varphi)\right] \\ & s.t. \quad x_{i,\omega} = T_i(x_i, u_i, \omega). \end{aligned} \tag{7}$$

3.1. Asset Management Example

We now give the formulation of the Asset Management problem from [12] as a CSO problem. Since the transition and objective functions are linear for each of the stages, we can write the stage problems using the notation of Linear Programming.

Note:. by giving the stage graph transition matrix, and the distributions of Ω, we don't need to include the future cost terms in the stage definition as these are implied. Therefore, we only include the $C_i(x_i, u_i, \omega)$ term in the objective.

Stage Graph. There are four sequential stages: `Today`; `Year 5`; `Year 10`; and `Horizon`. `Today` is a deterministic Decision-Only stage. The next three stages are Hazard-Decision problems. All arcs have probability one.

$P_{i,j} =$

	Today	Year 5	Year 10	Horizon
Root	1	0	0	0
Today	0	1	0	0
Year 5	0	0	1	0
Year 10	0	0	0	1
Horizon	0	0	0	0

(8)

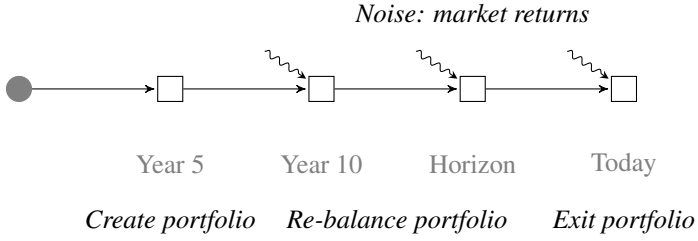

Figure 14: Asset Management Structure: graph

Today (Decision-Only).

$$\begin{aligned} \mathbf{D}_{\text{today}}\left(\begin{bmatrix} \bar{x}^s \\ \bar{x}^b \end{bmatrix}\right): \quad & \min \quad 0 \\ & \text{s.t.} \quad x^s + x^b = 55 + \bar{x}^s + \bar{x}^b \\ & \qquad x^s, x^b \geq 0, \end{aligned} \tag{9}$$

where $\bar{x}^s$ is the value of stocks (in dollars) held at the start of the stage, x^s is the value of stocks (in dollars) held at the end of the stage, $\bar{x}^b$ is the value of bonds (in dollars) held at the start of the stage, and x^b is the value of bonds (in dollars) held at the end of the stage. The control variables (value of stocks and bonds to buy) are implicit in this formulation.

The first stage has the initial solution $(\bar{x}^s, \bar{x}^b) = (0, 0)$.

Year 5 and Year 10 (Hazard-Decision).

$$\mathbf{HD}_{\text{rebalance}}\left(\begin{bmatrix} \bar{x}^s \\ \bar{x}^b \end{bmatrix}, \begin{bmatrix} \omega^s \\ \omega^b \end{bmatrix}\right): \quad \begin{aligned} \min_{\Delta} \quad & 0 \\ \text{s.t.} \quad & \omega^s \bar{x}^s + \omega^b \bar{x}^b = x^s + x^b \\ & x^s = \bar{x}^s + \Delta \\ & x^s, x^b \geq 0, \end{aligned} \tag{10}$$

where ω^s is the increase in the value of the stocks during the stage, and ω^b is the increase in the value of the bonds during the stage. The control variable Δ is the value of stocks to buy.

If these stages were Decision-Hazard, we could add the constraint that Δ is non-anticipative.

Horizon (Hazard-Decision).

$$\mathbf{HD}_{\text{horizon}}\left(\begin{bmatrix} \bar{x}^s \\ \bar{x}^b \end{bmatrix}, \begin{bmatrix} \omega^s \\ \omega^b \end{bmatrix}\right): \quad \begin{aligned} \min \quad & 4u - v \\ \text{s.t.} \quad & \omega^s \bar{x}^s + \omega^b \bar{x}^b + u - v = 80 \\ & u, v \geq 0, \end{aligned} \tag{11}$$

where u is the quantity by which the portfolio is less than the target ($80), and v is the quantity by which the portfolio exceeds the target.

Exogenous Information. Stages Year 5, Year 10, and Horizon have the following probability set for ω of:

$$(\omega^s, \omega^b) = \begin{cases} (-1.25, -1.14) & w.p.\ 0.5, \\ (-1.06, -1.12) & w.p.\ 0.5. \end{cases} \tag{12}$$

4. The SOF File-Format

In this section, we detail a new file-format (which we call the Stochastic Optimization Format (`.sof`)) for representing a subset of CSO problems. Namely, those problems with discrete random variables supported by known probabilities. However, before we introduce the format, we make a brief digression to describe a way of parameterizing ASCII text files used to represent mathematical programs (such as the LP, MPS, and NL file-formats).

4.1. Parameterizing Mathematical Optimization File-Formats

A common paradigm in mathematical optimization is to repeatedly solve many optimization models with changing parameters. For example, in traditional Linear Programming, one may want to solve the same LP with differing right-hand-side vectors. In the MPS file-format, this is done by introducing the ability to specify multiple right-hand-side vectors. However, we propose that this instead be done by providing a base file, along with a parameters file. The parameters

file is a JSON file that contains a single array of JSON objects. Each object contains a set of key-value pairs which are iteratively used to find-and-replace occurrences of the key in the base file with the value according the the GNU Bash specification for `sed`: `sed s/[key]/[value]/g`. Therefore, valid key-value pairs are any pairs representable as valid JSON, and valid inputs to `sed`. The user should not rely on the the key-value pairs to be iterated over in any order, and so no key or value should be a substring of any other. We recommend as convention, that parameter keys begin and end with the exclamation mark (ASCII-33).

4.2. Description of the `.sof` format

A `.sof` file is a JSON file composed of one object with the following fields. By convention, SOF files are named with a `.sof.json` extension so that they open automatically in a program that can recognize JSON (if one is present).

4.2.1. `"states"`

The `states` field must be a JSON object. The key in each key-value pair in the object corresponds to the name of one single-dimensional state variable. It must be unique. The value corresponds to the initial value of the state variable at the root node. It must be numeric and feasible.

States can be named any valid ASCII string that does not conflict with a reserved name in the file-format used to store the stage problems.

4.2.2. `"stages"`

The `stages` field is a JSON object that contains a key-value pair for each stage in the stage graph. The key corresponds to the name of the stage. The key `"ROOT"` is a reserved key. The value is a JSON object with two fields

1. `"fixed-variables"`: a JSON array of strings that refer to the variables that are non-anticipative in a Decision-Hazard stage. If `fixed-variables` is empty, the stage is assumed to be Hazard-Decision.
2. `"realizations"`: a JSON array containing one JSON object for each realization of the random variable in the stage). Each object in the array has three fields. Those fields are:
   (a) `"probability"`: a numeric value between 0.0 and 1.0 that corresponds to the probability of sampling that realization of the random variable.
   (b) `"basemodel"`: an ASCII string the corresponds to the name of a key in the JSON object described in section 4.2.4.
   (c) `"parameters"`: a JSON object comprised of a set of parameters that are used to modify the base model according to the procedure outlined in section 4.1.
   (d) `"fixed-variables"`: a JSON object with a key-value pair for each fixed variable mentioned above. The key should correspond to the string in the `fixed-variables` array above, and the value should be a string that refers to the name of the non-anticipative variable in the base model.

4.2.3. `"edges"`

The `edges` field is used to store the edges in the stage graph, along with their corresponding transition probabilities. It is an array of arrays, where each array in the main array is composed of three elements. The first is the parent stage, the second is the child stage, and the third element is the probability of transitioning from the parent to the child (decimal value between 0.0 and

1.0). The stage name `"ROOT"` is a reserved name, and there must be at least one array in `edges` that contains the string `"ROOT"`. Further, it must appear in the first element, and not the second element.

4.2.4. "basemodels"

1. `"format"`: the file extension that is used to determine how the model string should be interpreted. Likely formats are `.lp`, `.mps`, and `.nl`.
2. `"states"`: a JSON object that contains one key-value pair for each state in the JSON object described in section 4.2.1. Each value must be a JSON object with two key-value pairs. One key must be `"in"`. The value of the `in` pair must correspond to the name of the incoming state variable in the base model. If the model file is a NL file, the value will be the integer index of the variable in the NL file. The other key must be `"out"`. The value of the `out` pair must correspond to the name of the outgoing state variable in the model file.
3. `"parameters"`: a JSON object contains a set of parameters that are used to modify the model. The value of each key-value pair is taken as the default value. Parameter values specified in the `stage` object (section 4.2.2) override the default values.
4. `model`: a string containing an ASCII representation of the base model.

4.2.5. Additional fields

The JSON file may also contain other fields (for example `author`, `description`, or `license`), which should be in human-readable form for explanatory purposes only.

4.3. Compression

The SOF file was not designed to be concise. Indeed, with large realistic model instances, the resulting files may be very large. However, the filesize can be reduced significantly if compressed using an algorithm such as gzip. We recommend that implementations of SOF readers and writers include a facility to inflate and deflate such files.

4.4. The Asset Management Example

In the Appendix, we give the files for the Asset Management example from section 3.1 in the SOF file-format, as well as the SMPS format (from [2]). Although the uncompressed version of the SOF file is significantly larger than the SMPS files (4574 bytes for the SOF, compared to 1289 bytes), the compressed version is smaller (778 bytes). One reason for this is that the SMPS files contain a lot of implied assumptions (for example, the entries in the core file are in temporal order). This reduces the quantity of information that needs to be conveyed in the SMPS files. Furthermore, the SOF file is considerably more human-readable due to the usage of the LP file-format, rather than MPS file-format. The SOF format also allows arbitrary meta-data to be stored with the model so we can acknowledge things such as the original author of the model.

5. Open-Source Implementation and Test-Set

Goals

1. collate a variety of examples
2. provide an interface to `https://github.com/odow/SDDP.jl`
3. benchmark

6. Conclusion

In this paper we have presented a graphical representation of the sequential decision making process that clarifies the timing in which decisions are made and uncertainty is revealed. We then propose a file-format that is built on-top of the widely accepted LP and .NL file-formats. This alleviates many of the issues with the rigidity of the SMPS file-format, but is not able to represent continuous random variables. The `.sof` file-format is capable of representing any CSO problem with sequential decision making and random variables that take discrete values.

Acknowledgements

Many people have contributed, both directly and indirectly to these ideas. They include Andy Philpott, Tony Downward, and Andrew Mason from The University of Auckland; Vincent Leclère, François Pacaud, Tristan Rigaut, and Henri Gerard from École des Ponts; and Benoît Legat from Université catholique de Louvain; Joaquim Dias Garcia and Camila Metello from PSR; Zachary Sunberg from Stanford; and Jordan Jalving from University Wisconsin-Madison.

References

[1] D. Klingman, A. Napier, J. Stutz, NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems, management science 20 (5) (1974) 814–821.
URL http://pubsonline.informs.org/doi/abs/10.1287/mnsc.20.5.814

[2] H. I. Gassmann, B. Kristjansson, The SMPS format explained, IMA Journal of Management Mathematics 19 (4) (2007) 347–377. doi:10.1093/imaman/dpm007.
URL https://academic.oup.com/imaman/article-lookup/doi/10.1093/imaman/dpm007

[3] J. R. Birge, M. A. Dempster, H. I. Gassmann, E. Gunn, A. J. King, S. W. Wallace, A standard input format for multiperiod stochastic linear programs, IIASA Working Paper, WP-87-118, IIASA, Laxenburg, Austria (1987).
URL http://pure.iiasa.ac.at/2934

[4] Holmes, D., A (PO)rtable (S)tochastic programming (T)est (S)et (POSTS).
URL http://users.iems.northwestern.edu/~jrbirge/html/dholmes/post.html

[5] K. A. Ariyawansa, A. J. Felt, On a New Collection of Stochastic Linear Programming Test Problems, INFORMS Journal on Computing 16 (3) (2004) 291–299. doi:10.1287/ijoc.1030.0037.
URL http://pubsonline.informs.org/doi/10.1287/ijoc.1030.0037

[6] S. Ahmed, R. Garcia, N. Kong, L. Ntaimo, G. Parija, F. Qiu, S. Sen, SIPLIB.
URL http://www2.isye.gatech.edu/~sahmed/siplib/

[7] J. Linderoth, A. Shapiro, S. Wright, The empirical behavior of sampling methods for stochastic programming, Annals of Operations Research 142 (1) (2006) 215–241. doi:10.1007/s10479-006-6169-8.
URL http://link.springer.com/10.1007/s10479-006-6169-8

[8] V. Zverovich, C. I. Fábián, E. F. D. Ellison, G. Mitra, A computational study of a solver system for processing two-stage stochastic LPs with enhanced Benders decomposition, Mathematical Programming Computation 4 (3) (2012) 211–238. doi:10.1007/s12532-012-0038-z.
URL http://link.springer.com/10.1007/s12532-012-0038-z

[9] W. B. Powell, Clearing the Jungle of Stochastic Optimization, in: A. M. Newman, J. Leung, J. C. Smith, H. J. Greenberg (Eds.), Bridging Data and Decisions, INFORMS, 2014, pp. 109–137.
URL http://pubsonline.informs.org/doi/abs/10.1287/educ.2014.0128

[10] W. B. Powell, A Unified Framework for Optimization under Uncertainty TutORials in Operations Research, in: Optimization Challenges in Complex, Networked and Risky Systems, 2016, pp. 45–83.
URL http://castlelab.princeton.edu/Papers/Powell-UnifiedFrameworkforStochasticOptimization_April162016.pdf

[11] R. Fourer, H. I. Gassmann, J. Ma, R. K. Martin, An XML-based schema for stochastic programs, Annals of Operations Research 166 (1) (2009) 313–337. doi:10.1007/s10479-008-0419-x.
URL http://link.springer.com/10.1007/s10479-008-0419-x

[12] J. R. Birge, F. Louveaux, Introduction to Stochastic Programming, Springer Series in Operations Research and Financial Engineering, Springer New York, New York, NY, 2011, dOI: 10.1007/978-1-4614-0237-4.
URL http://link.springer.com/10.1007/978-1-4614-0237-4

Appendix

SMPS Files for Asset Management

asset_management.cor

```
NAME       Asset Mgt
ROWS
 N WEALTH
 E BUDGET
 E BAL1
 E BAL2
 E BAL3
COLUMNS
    STOCK1    BUDGET     1.00     BAL1     -1.15
    BONDS1    BUDGET     1.00     BAL1     -1.13
    STOCK2    BAL1       1.00     BAL2     -1.15
    BONDS2    BAL1       1.00     BAL2     -1.13
    STOCK3    BAL2       1.00     BAL3     -1.15
    BONDS3    BAL2       1.00     BAL3     -1.13
    SHORT     BAL3      -1.00     WEALTH    1.00
    OVER      BAL3       1.00     WEALTH   -1.00
RHS
    RHS       BUDGET     55.00    BAL3     80.00
ENDATA
```

asset_management.tim

```
TIME Asset Mgt
PERIODS
    STOCK1    BUDGET    TODAY
    STOCK2    BAL1      YEAR_5
    STOCK3    BAL2      YEAR_10
    SHORT     BAL3      HORIZON
ENDATA
```

asset_management.sto

```
STOCH Asset Mgt
SCEN
SC  SCEN_1     'ROOT'     0.125     TODAY
    STOCK1     BAL1      -1.25
    BONDS1     BAL1      -1.14
    STOCK2     BAL2      -1.25
    BONDS2     BAL2      -1.14
    STOCK3     BAL3      -1.25
    BONDS3     BAL3      -1.14
SC  SCEN_2     SCEN_1     0.125     HORIZON
    STOCK3     BAL3      -1.06
    BONDS3     BAL3      -1.12
SC  SCEN_3     SCEN_1     0.125     YEAR_10
    STOCK2     BAL2      -1.06
    BONDS2     BAL2      -1.12
```

```
    STOCK3     BAL3      -1.25
    BONDS3     BAL3      -1.14
SC  SCEN_4     SCEN_3     0.125     HORIZON
    STOCK3     BAL3      -1.06
    BONDS3     BAL3      -1.12
SC  SCEN_5     SCEN_1     0.125     YEAR_5
    STOCK1     BAL1      -1.06
    BONDS1     BAL1      -1.12
    STOCK2     BAL2      -1.25
    BONDS2     BAL2      -1.14
    STOCK3     BAL3      -1.25
    BONDS3     BAL3      -1.14
SC  SCEN_6     SCEN_5     0.125     HORIZON
    STOCK3     BAL3      -1.06
    BONDS3     BAL3      -1.12
SC  SCEN_7     SCEN_5     0.125     YEAR_10
    STOCK2     BAL2      -1.06
    BONDS2     BAL2      -1.12
    STOCK3     BAL3      -1.25
    BONDS3     BAL3      -1.14
SC  SCEN_8     SCEN_7     0.125     HORIZON
    STOCK3     BAL3      -1.06
    BONDS3     BAL3      -1.12
ENDATA
```

CSO File for Asset Management

asset_management.sof

```
{
    "author":"Oscar Dowson",
    "description":"The Asset Management problem taken from R. Birge,
        F. Louveaux,  Introduction to Stochastic Programming,
        Springer Series in Operations Research and Financial
        Engineering, Springer New York, New York, NY, 2011.",
    "states":{"stocks": 0, "bonds": 0},
    "stages":{
        "today": {
            "fixed-variables": [],
            "realizations": [
                {
                    "probability": 1.0,
                    "basemodel": "today",
                    "parameters": {},
                    "fixed-variables": {}
                }
            ]
        },
        "year_5": {
```

```
            "fixed-variables": [],
            "realizations": [
                {
                    "probability": 0.5,
                    "basemodel": "rebalance",
                    "parameters": {
                        "!STOCKRETURN!": 1.25,
                        "!BONDRETURN!": 1.14
                    },
                    "fixed-variables": {}
                },
                {
                    "probability": 0.5,
                    "subproblem": "rebalance",
                    "parameters": {
                        "!STOCKRETURN!": 1.06,
                        "!BONDRETURN!": 1.12
                    },
                    "fixed-variables": {}
                }
            ]
        },
        "year_10": {
            "fixed-variables": [],
            "realizations": [
                {
                    "probability": 0.5,
                    "basemodel": "rebalance",
                    "parameters": {
                        "!STOCKRETURN!": 1.25,
                        "!BONDRETURN!": 1.14
                    },
                    "fixed-variables": {}
                },
                {
                    "probability": 0.5,
                    "basemodel": "rebalance",
                    "parameters": {
                        "!STOCKRETURN!": 1.06,
                        "!BONDRETURN!": 1.12
                    },
                    "fixed-variables": {}
                }
            ]
        },
        "horizon": {
            "fixed-variables": [],
```

```
            "realizations": [
                {
                    "probability": 0.5,
                    "basemodel": "horizon",
                    "parameters": {
                        "!STOCKRETURN!": 1.25,
                        "!BONDRETURN!": 1.14
                    },
                    "fixed-variables": {}
                },
                {
                    "probability": 0.5,
                    "basemodel": "horizon",
                    "parameters": {
                        "!STOCKRETURN!": 1.06,
                        "!BONDRETURN!": 1.12
                    },
                    "fixed-variables": {}
                }
            ]
        }
    },
    "edges": [
        ["ROOT", "today", 1.0],
        ["today", "year_5",1.0],
        ["year_5", "year_10",1.0],
        ["year_10", "horizon",1.0]
    ],
    "basemodels": {
        "today": {
            "format": ".lp",
            "states":{
                "stocks": {"in": "STOCK_IN", "out": "STOCK_OUT"},
                "bonds": {"in": "BONDS_IN", "out": "BONDS_OUT"}
            },
            "parameters": {},
            "model": "Minimize
                obj:
                Subject To
                C1: 1 STOCK_OUT + 1 BONDS_OUT == 55.0
                Bounds
                STOCK_IN free
                BONDS_IN free
                0 <= STOCK_OUT <= +inf
                0 <= BONDS_OUT <= +inf
                General
                Binary
```

```
            End"
    },
    "rebalance": {
        "format": ".lp",
        "states":{
            "stocks": {"in": "STOCK_IN", "out": "STOCK_OUT"},
            "bonds": {"in": "BONDS_IN", "out": "BONDS_OUT"}
        },
        "parameters": {"!STOCKRETURN!": 0.0, "!BONDRETURN!": 0.0},
        "model": "Minimize
            obj:
            Subject To
            C1: -!STOCKRETURN! STOCK_IN - !BONDRETURN! BONDS_IN
                + 1 STOCK_OUT + 1 BONDS_OUT == 0.0
            Bounds
            STOCK_IN free
            BONDS_IN free
            0 <= STOCK_OUT <= +inf
            0 <= BONDS_OUT <= +inf
            General
            Binary
            End"
    },
    "horizon": {
        "format": ".lp",
        "states":{
            "stocks": {"in": "STOCK_IN", "out": "STOCK_OUT"},
            "bonds": {"in": "BONDS_IN", "out": "BONDS_OUT"}
        },
        "parameters": {"!STOCKRETURN!": 0.0, "!BONDRETURN!": 0.0},
        "model": "Minimize
            obj: -1 OVER + 4 SHORT
            Subject To
            C1: -!STOCKRETURN! STOCK_IN - !BONDRETURN! BONDS_IN
                - 1 OVER + 1 SHORT == 80.0
            Bounds
            STOCK_IN free
            BONDS_IN free
            0 <= STOCK_OUT <= +inf
            0 <= BONDS_OUT <= +inf
            0 <= OVER <= +inf
            0 <= SHORT <= +inf
            General
            Binary
            End"
    }
}
```

```
}
```