OPTIMA 96

Mathematical Optimization Society Newsletter

Note from the Editors

Dear MOS members,
In the main article of this Optima issue in your hands, Warren B. Powell (Princeton University) deals with the question of how a general model of optimization problems involving uncertainties could look like – a model that would allow for describing and discussing models from an entire spectrum of areas such as stochastic programming, dynamic programming, and robust optimization in a unified way. In the follow-up discussion column, Andrzej Ruszczynski (Rutgers University) analyzes four aspects of Powell's proposal raising in particular the question to what extent the proposed unified model suitably covers all relevant aspects.

We would be happy if these contributions to Optima started a fruitful debate within the community. In this spirit, we already have a short rebuttal by Warren Powell that you will find immediately after Andrzej Ruszczynski's remarks. We hope that this issue of our newsletter initiates discussions about the "right way to state optimization problems under uncertainty", e.g., at the upcoming ISMP 2015 in Pittsburgh.

In case you have not yet made plans for ISMP 2015, you will want to do so after having a look at the list of outstanding plenary and semiplenary speakers whose short biographies we also include in this issue.

With our best wishes for the new year! Sam Burer, Co-Editor
Volker Kaibel, Editor
Jeff Linderoth, Co-Editor

Warren B. Powell

Bridging the Fields of Stochastic Optimization

While there is almost universal acceptance of the canonical form for a deterministic optimization problem, the study of sequential stochastic optimization problems has become fragmented into a diverse set of communities which differ in terms of applications and notation, modeling and algorithmic strategies, and even the types of questions that are considered publishable contributions. It is quite common that authors cannot agree on an objective function, or even what they are optimizing over!

Perhaps the most visible debate in operations research has been between the frameworks offered under the headings of "dynamic programming," "stochastic programming," and "robust optimization," with different research groups promoting the strengths of each approach. However, these arguments quickly blend with methods that have evolved using names such as reinforcement learning, stochastic search, simulation optimization, and optimal control. Often overlooked are the communities working on online versions of these problems using the terminology of multiarmed bandit problems.

An example of the current state of affairs is captured by a comment made by a well-intentioned referee:

> One of the main contributions of the paper is the demonstration of a policy-based modeling framework for transportation problems with uncertainty. However, it could be argued that a richer modeling framework already exists (multi-stage stochastic programming) that does not require approximating the decision space with policies.

The confusion is leading to claims that one method is "better" than another method. Many authors have criticized dynamic programming because of the "curse of dimensionality"; stochastic dual dynamic programming (SDDP) has been described as a "gold standard"; robust optimization overcomes the curse of dimensionality (one colleague insisted to me that it "always works"); and a host of heuristic policies have been promoted because they enjoy some sort of bound.

In our experience, none of these statements is always true, and we believe that almost every proposed methodology is best (or at least very good) for at least *some* problem. What is missing is an appreciation that this discussion is effectively a heuristic search for the best policy to solve an optimization problem that is rarely formulated properly. The goal of this article is to articulate and formalize this process.

In this article, we are going to argue for a particular canonical form for sequential stochastic optimization problems that unifies dynamic programming, stochastic programming, and robust optimization, along with every other approach used to solve this problem class. Our goal is not to claim that any method is better than any other method, but rather to agree on the problem that is being solved so that we have a common basis for comparing solutions. Finally, we would like to get an agreement on precisely what it is that we are looking for.

1 A canonical model for sequential stochastic optimization

If we are solving a deterministic version of a time-staged problem, we might write the problem as the following linear model

$$\min_{x_0,\dots,x_T} \sum_{t=0}^{T} c_t x_t \tag{1}$$

Contents of Issue 96 / December 2014

subject to

$$A_t x_t = R_t, \qquad t = 0, \dots, T, \tag{2}$$
$$R_{t+1} = R_t + B_t x_t, \qquad t = 0, \dots, T-1, \tag{3}$$
$$x_t \le u_t, \qquad t = 0, \dots, T, \tag{4}$$
$$x_t \ge 0, \qquad t = 0, \dots, T. \tag{5}$$

Thus, we have expressed a decision vector, an objective function, and a set of constraints.

A generic, static stochastic optimization problem is widely written in a form that is close to

$$\min_{x \in \mathcal{X}} \mathbb{E} F(x, W), \tag{6}$$

where W is a random variable (or random vector), $\mathbb{E}$ is the expectation operator over the set of possible outcomes of W, and $\mathcal{X}$ is the feasible region.

But what happens when we have a sequential problem? Here, we have to deal with a sequence of decisions x_t and information W_t, starting with an initial state S_0. An author writing the problem as a dynamic program might start by writing

$$V_t(S_t) = \min_{x_t} \Big(C(S_t, x_t) + \gamma \sum_{s' \in \mathcal{S}} p(s'|S_t, x_t) V_{t+1}(s') \Big), \tag{7}$$

where γ is a discount factor, $\mathcal{S}$ is our state space, $V_t(s)$ is the value of being in state s, and $p(s'|S_t, x_t)$ is the probability of transitioning to state s' if we are in state S_t and take decision x_t (more typically written as action a_t).

Since we know that we cannot compute (7) for anything other than a toy problem, we can turn to stochastic programming ([3], [6]), where we might face the problem of solving

$$\min_{x_0, (x_t(\omega), 0 < t \le H), \forall \omega \in \Omega} \Big(c_0 x_0 + \sum_{\omega \in \Omega} p(\omega) \sum_{t=1}^{H} c_t(\omega) x_t(\omega) \Big), \tag{8}$$

subject to constraints that capture not only stochastic versions of (2)–(5), but also nonanticipativity constraints that ensure that we are not making decisions using information that is not available yet. We formulate (8) over a planning horizon H that is typically less than the model horizon T.

A simpler alternative appears to be robust optimization [1], which replaces an approximation of the expectation with optimization over an uncertainty set (we revisit this below).

We note, however, that while (1)–(5) represents a deterministic, optimization *model*, neither (7) nor (8) represents what could be called the stochastic counterpart. Oddly, we cannot find any evidence of the kind of canonical form for a sequential stochastic optimization problem enjoyed by the deterministic community (with notable exceptions in pocket communities such as multiarmed bandits).

We argue (as we have in [5, Chapter 5]) that there are five elements to a sequential, stochastic optimization problem:

1. *The state* S_t. This is all the information needed to compute our model from time t onward, which means it has to include the information in the cost function, decision function and transition function for all remaining time periods. The state consists of the resource or physical state R_t (inventories, state of machines), exogenous information I_t (basically any data we need that is not in the resource/physical state), and belief/knowledge state K_t (probability distributions giving our belief about unobservable parameters).
2. *Decisions* x_t. These are the controllable quantities. We do not specify how we are going to make a decision, but we assume that there will be a decision function (policy) which we denote by $X_t^\pi(S_t)$, $\pi \in \Pi$, which produces a feasible vector x_t given the information in S_t. Here, "π" carries information about the structure of the function and any tunable parameters. We can think of Π as some arbitrary set of allowable functions (policies), but below, we are going to refine this to a well-defined set of computable functions.
3. *Exogenous information* W_t. This is new information that is learned between time $t-1$ and t. We assume that any variable indexed by t is known at time t (that is, it is "$\mathcal{F}_t$-measurable" in the parlance of the probability community). We note that the information may depend on both the state S_t and the decision x_t.
4. *Transition function*. This is the set of equations that takes us from S_t to S_{t+1} which we write using

$$S_{t+1} = S^M(S_t, x_t, W_{t+1}). \tag{9}$$

While this term is not generally used in operations research, this is widely known as the system model, plant model, or simply "model" in the engineering controls community, where the term is standard. The transition function includes both the linear equations that might be used to model the controlled evolution of physical resources, as well as updates due to the arrival of exogenous information (in stochastic programming, this is how scenario trees are constructed).
5. *The objective function*. Here we formally state our desire to find the best policy. This is normally written

$$\min_{\pi \in \Pi} \mathbb{E}^\pi \sum_{t=0}^{T} C(S_t, X_t^\pi(S_t)), \tag{10}$$

where $C(S_t, x_t)$ is the cost at time t if we are in state S_t and make decision x_t. Note that the transition function (9) is imbedded in (10).

We do not have to use an expectation. We might instead use a risk measure $\varrho(\cdot)$ which could be a quantile or a function of a random variable that penalizes outcomes that are perhaps above average, giving us the objective function

$$\min_{\pi \in \Pi} \varrho\Big(\sum_{t=0}^{T} C\big(S_t, X_t^\pi(S_t)\big) \Big). \tag{11}$$

We might even want to focus on the worst outcome, which is the objective used in robust optimization, a topic we revisit below. The choice of objective is, of course, up to the modeler. Our job, as algorithms specialists, is to solve the model that has been presented to us.

We argue that these five elements represent what should be the canonical form for any sequential ("multistage") stochastic decision problem. We observe here that equation (10) represents a dynamic program; we sometimes have to remind ourselves that a dynamic program is a *problem*, not an algorithm (or solution).

Equation (10) is the stochastic version of the objective function in equation (1) for a deterministic optimization problem, assuming we are using expectations (otherwise we might use (11)). Equation (3) is the deterministic analog of our transition function, while equations (2), (4) and (5) are the constraints that have to be observed by our policy $X_t^\pi(S_t)$. Yet, while virtually *every* statement of a deterministic optimization model will write out an objective function such as (1), it is actually quite rare to see equation (10) (or (11)) written explicitly.

We claim that every model of a sequential stochastic optimization problem should include the objective function using some form

of (10) or (11). We refer to this objective function (using whatever form is appropriate for a particular application) as the *base model*. This will help distinguish it from an important class of policies that use a *lookahead model*, which we describe below.

In practice, we cannot compute the expectation (or risk measure), so we might simulate a series of N sample paths $(\omega^n)_{n=1}^N$. If these occur with equal likelihood, this means that we estimate the value of a policy $X_t^\pi(S_t)$ using

$$\bar{F}^\pi = \frac{1}{N}\sum_{n=1}^{N}\sum_{t=0}^{T} C\Big(S_t(\omega^n), X_t^\pi\big(S_t(\omega^n)\big)\Big) \tag{12}$$

where $S_{t+1}(\omega^n) = S^M(S_t(\omega^n), X_t^\pi(S_t(\omega^n)), W_{t+1}(\omega^n))$. It is quite common that authors will view (12) as a "simulation" rather than an approximation of the objective function. It is not unusual to see a paper describe the process of simulating a policy without actually writing the objective function. We feel it is important to explicitly recognize the base model as the objective function, just as we write out (1) as the objective function for our deterministic model.

We note that the objectives (10) (or (11)) are mathematically equivalent to (6), which is widely used by the static stochastic optimization community, where the search is over a real-valued vector. However, if we have a sequential decision problem, we need to find the best decision function (policy), which presumably can be characterized by π that captures the structure of the function and any tunable parameters (which may themselves be time-dependent). The preferred approach in the research literature (by a wide margin) is to simply pick a class of policies, after which it is generally possible to tune parameters that characterize the class (although this last step is not included in many papers).

By writing our objective function as (10), we can resolve the question of what we are looking for. In deterministic optimization (and static stochastic optimization), we are searching for the best *decision* (typically a real-valued vector). In sequential stochastic optimization, we are searching for a *function*, which is our policy for making decisions. The problem is that while we have an extensive library of algorithms for finding the best decision, we lack a similar framework for searching for policies. We address this challenge next.

2 Designing a policy

A policy is a mapping from a state S_t to a feasible action x_t, which means it is some type of function. Finding the best policy in (10) requires searching over a class of functions. The problem is, we just do not know how to translate this to a practical algorithm.

We begin by providing a characterization of an optimal policy, if only to serve as a reference point. We then propose what we feel are four fundamental classes of policies which appear to form the basis of every solution strategy we have encountered in practice or in the literature.

2.1 The optimal policy

We can always express an optimal policy by writing

$$X_t^*(S_t) = \arg\min_{x_t}\Big(C(S_t,x_t) + \min_{\pi\in\Pi}\mathbb{E}\Big\{\sum_{t'=t+1}^{T} C\big(S_{t'}, X_{t'}^\pi(S_{t'})\big)\big|S_t,x_t\Big\}\Big), \tag{13}$$

where $S_{t'+1} = S^M(S_{t'}, x_{t'}, W_{t'+1})$ (note that this only works if we are using the expectation form of the objective in (10)). Not surprisingly, this is computationally meaningless; we do not know how to compute the minimization over policies, and even if this was well defined, we generally cannot compute the imbedded expectation.

Because the optimization over policies can look mysterious, the stochastic programming community often writes

$$X_t^*(S_t) = \arg\min_{x_t}\Big(C(S_t,x_{tt}) + \min_{(x_{tt'}(\omega),t<t'\le T),\forall\omega\in\Omega_t}\mathbb{E}\Big\{\sum_{t'=t+1}^{T} C\big(S_{tt'}, x_{tt'}(\omega)\big)\big|S_t\Big\}\Big). \tag{14}$$

Instead of writing a policy $X_{t'}^\pi(S_{t'})$, we are writing $x_{tt'}(\omega)$, which means that the decision at time t' depends on the sample path that we are following. The problem is that when we specify ω, it means we are identifying the *entire* sample path, from $t=0$ up to $t=T$, which is like being able to know the future. For this reason, it is necessary to introduce *nonanticipativity constraints* (see [3]).

A better way of writing (14) is to write $x_{tt'}(S_{tt'})$ instead of $x_{tt'}(\omega)$, which avoids the need to deal with nonanticipativity constraints, but it still means figuring out the functional form for $x_{tt'}(S_{tt'})$ (this is basically our policy $X_{t'}^\pi(S_{tt'})$). Either way, equation (14) is not computable.

There are five strategies for approximating the lookahead model. These are:

1. *Limiting the horizon*. Instead of optimizing from t to T, we optimize over a planning horizon t to $t+H$.
2. *Stage aggregation*. It is very common to approximate the sequencing of information and decisions over the horizon with a two-stage model, where all random information is revealed at once after x_t is determined.
3. *Information approximation*. The most common approximations of the information model is outcome aggregation or Monte Carlo sampling, where the full set of outcomes Ω_t is replaced with a sample $\hat{\Omega}_t$. These are referred to as scenarios in the stochastic programming community. The strategy used by robust optimization replaces Ω_t with an uncertainty set, and then replaces the expectation (a form of averaging operator) with a maximization over the uncertainty set (more on this below).
4. *Discretization*. We may use a coarse discretization of states, time and decisions to simplify the lookahead model.
5. *Dimensionality reduction*. It is surprisingly common to hold some information constant within the lookahead model. For example, we may have a set of forecasts $f_{tt'}$ for $t' = t+1,\dots,t+H$. But this ignores the reality that at time $t+1$, the forecasts will change. This has the effect of dropping forecasts from the state variable within the lookahead model (such variables are known as *latent variables*).

2.2 The four classes of policies

Our tour through stochastic optimization papers over the last 30 years led to the conclusion that there are four fundamental classes of policies if we looked just at what people were actually using (that is, computable policies). These are:

1. *Policy function approximations (PFAs)*. These are analytic functions that map states to actions, without using an imbedded optimization problem. These might be discrete lookup tables (when at this node, turn left), parametric functions (such as (s,S) inventory policies), or statistical models, such as
$$X^\pi(S_t|\theta) = \theta_0 + \theta_1 S_t + \theta_2 S_t^2. \tag{15}$$
2. *Policies based on robust cost function approximations (CFAs)*. To illustrate a CFA, imagine starting with a simple myopic policy that we can write
$$X_t^\pi(S_t) = \arg\min_{x\in\mathcal{X}_t} C(S_t,x).$$

Not surprisingly, this would rarely work well in practice. However, there are problems where a slightly modified cost function might work quite well. One class of approximations looks like

$$X_t^\pi(S_t|\theta) = \arg\min_{x\in\mathcal{X}_t} \big(C(S_t,x) + \sum_{f\in\mathcal{F}} \theta_f \phi_f(S_t,x)\big). \quad (16)$$

Here, $(\phi_f(S_t,x))_{f\in\mathcal{F}}$, is a set of *basis functions* (as they are known in the approximate dynamic programming community) which might be of the form S_tx, S_tx^2, x, x^2, which serves as a type of correction term. However, there are other problems where we make direct changes to the cost function itself, or perhaps the constraints (e.g., imposing a minimum inventory level). We can represent this class of policies more broadly by writing

$$X_t^\pi(S_t|\theta) = \arg\min_{x\in\mathcal{X}_t^\pi(\theta)} \bar{C}^\pi(S_t,x|\theta). \quad (17)$$

where $\bar{C}_t^\pi(S_t,x|\theta)$ is some sort of parametric approximation of the costs, while $\mathcal{X}_t^\pi(\theta)$ might be a modified set of constraints (for example, we might introduce schedule slack in a scheduling problem, or buffer stocks in an inventory problem). Here, we would let π carry the information about the structure of the approximation (such as the basis functions in equation (16)) and we let θ capture all tunable parameters.

3. *Policies based on value function approximations (VFAs)*. These are the policies most often associated with dynamic programming, and are written

$$X_t^\pi(S_t|\theta) = \arg\min_{x\in\mathcal{X}_t} \big(C(S_t,x) + \mathbb{E}\{\overline{V}_{t+1}(S_{t+1}|\theta)|S_t\}\big). \quad (18)$$

This is what is generally known as approximate dynamic programming. A popular (if risky) strategy is to approximate the value function using a linear model, giving us

$$X_t^\pi(S_t|\theta) = \arg\min_{x\in\mathcal{X}_t} \Big(C(S_t,x) + \mathbb{E}\Big\{\sum_{f\in\mathcal{F}} \theta_f \phi_f(S_{t+1})\Big|S_t\Big\}\Big). \quad (19)$$

We could replace S_{t+1} with the post-decision state S_t^x and eliminate the expectation, producing a policy that looks cosmetically like our cost function approximation given in (16). However, the basis functions and regression parameters θ in (16) are completely different from those in (19), where they are being used to approximate the value of being in state S_{t+1} (or S_t^x), while in (16), the basis functions and θ bear no relation to a value function. In a CFA-based policy, the error correction term is not making any attempt whatsoever to approximate the value of being in a state. It is being tuned purely to produce a better policy.

4. *Lookahead policies*. Lookahead policies are based on approximations of the *model* from time t onward in equation (13). If we did not have to approximate the future, we would have an optimal policy. This approximation is called the *lookahead model*.

The simplest lookahead policy involves optimizing over a horizon H deterministically. Let $\tilde{x}_{tt'}$ represent the decision variables (this might be a vector) for time t' in the lookahead model that is being solved at time t (which determines the information content). Variables with tildes represent the lookahead model, so we do not confuse them with the base model. A deterministic lookahead policy might be written

$$X_t^\pi(S_t|\theta) = \arg\min_{\tilde{x}_{tt},\ldots,\tilde{x}_{t,t+H}} \sum_{t'=t}^{t+H} C(\tilde{S}_{tt'},\tilde{x}_{tt'}), \quad (20)$$

where θ captures the horizon and perhaps the discretization (or other approximations) used in forming the lookahead model.

The stochastic programming community approximates the future by using a sampled representation of the exogenous information process, producing a policy that might look like

$$X_t^\pi(S_t|\theta) = \arg\min_{\tilde{x}_{tt},(\tilde{x}_{tt'}(\tilde{\omega}),t<t'\le t+H),\tilde{\omega}\in\tilde{\Omega}_t} \tilde{c}_{tt}\tilde{x}_{tt} + \sum_{\tilde{\omega}\in\tilde{\Omega}_t} p(\tilde{\omega}) \sum_{t'=t+1}^{t+H} \tilde{c}_{tt'}(\tilde{\omega})\tilde{x}_{tt'}(\tilde{\omega}). \quad (21)$$

In this case, θ captures parameters such as the number of information stages, the number of scenarios per stage, temporal aggregation and discretization.

Below, we show how robust optimization (for sequential problems) not only approximates the information process (using an "uncertainty set"), but also replaces the expectation operator with a maximization over outcomes.

It is possible to form hybrid policies. For example, we might combine a low-dimensional policy function approximation (pump water into storage at night, release during the day) into a high-dimensional problem to plan energy generators. A lookahead policy might use value functions as a terminal approximation.

Three of our policies (PFAs, CFAs and VFAs) require approximating some function. Most approximation strategies can be categorized as lookup table, parametric or nonparametric (see [4] for an in-depth discussion). The lookahead policy, on the other hand, requires approximating the lookahead model itself. But all represent some form of approximation, and optimal policies are extremely rare. In fact, we often find people overlooking the fact that finding optimal solutions to stochastic lookahead models ("stochastic programs"), which can be a serious computational challenge, is generally *not* an optimal policy, and bounds on our solution of a stochastic program (which are often so large they cannot be solved optimally), are not bounds on the performance of the policy.

We represent a policy π as consisting of the class $C\in\mathcal{C}$ (its structural form) along with any tunable parameters θ that must belong to a set Θ^C. This means that the search over policies π in equation (10) means (a) searching over different classes $\mathcal{C}$ of policies and then (b) for a given class, searching over a set of tunable parameters Θ^C. Because hybrids can be very useful, we also have to consider searching over mixtures of classes.

2.3 Choosing a policy

Below we offer some guidance in terms of choosing a particular class of policies:

1. Policy function approximations work best for low-dimensional actions, where the structure of the policy is fairly obvious. (s,S) inventories are an easy example. Policy function approximations can also work well when the policy is a relatively smooth surface, allowing it to be approximated perhaps by a linear function (known in the literature as "affine policies") or locally linear functions. Neural networks are very popular in engineering applications.
2. Robust cost function approximations, which are typically deterministic models modified to handle uncertainty, work well for high-dimensional problems, where we can readily identify the change needed to produce a robust policy (such as including buffer stocks or schedule slack).
3. Value function approximations are particularly useful when the value of the future given a state is easy to approximate. This approach has proven particularly powerful when approximating problems that are convex in the decision variable, since convexity makes it possible to quickly approximate high dimensional functions with a relatively small number of observations.

4. Lookahead policies are particularly useful for time-dependent problems, and especially if there is a forecast available that evolves over time, an issue that is often overlooked in applications that use one of the other approximation strategies. Deterministic lookahead models work better than the research literature likes to acknowledge, especially for problems with random right hand sides, but struggle with random prices. Stochastic lookahead models offer a brute-force way to capture uncertainty, but should always be compared against a deterministic lookahead policy to make sure that the cost of solving a stochastic lookahead model is worth it. These comparisons should be done using equation (10), but this means that you *must* write a simulator (after all, this is the objective function!).

3 Robust optimization for sequential problems

Robust optimization has been primarily studied as a method for static stochastic optimization, where the expectation in (6) is replaced with a maximization problem (find the worst outcome) over an uncertainty set $\mathcal{W}$, giving us

$$\min_{x} \max_{w \in \mathcal{W}} F(x, w), \tag{22}$$

which transforms a stochastic optimization problem into a deterministic problem.

Equation (22) is an optimization *problem*, where the expectation has been replaced with a maximization. This approach is particularly appropriate in engineering design problems where a static design has to work under all possible conditions.

This idea has been extended to sequential applications, where the problem at time t is to solve

$$X_t^{\pi}(S_t|\theta) = \arg\min_{x_t \in \mathcal{X}_t} \max_{w \in \mathcal{W}_t(\theta)} \sum_{t'=t}^{t+H} C\big(S_{t'}(w), x_{t'}\big), \tag{23}$$

where $S_{t'+1}(w) = S^M(S_{t'}(w), x_{t'}, W_{t'+1}(w))$, and where $\mathcal{W}_t(\theta)$ is an uncertainty set constructed at time t, parameterized by θ. Equation (23) is a *policy* that has been suggested for sequential problems (see, for example, [2]). But it is a policy to solve what problem? Surprisingly, this is not stated explicitly in the robust optimization literature. For example, [2, Section 4.3], when describing their testing, state that they ran "hundreds of simulations [...] and compared the mean performance vis-à-vis the mean [...] outcome." This means that they use the robust policy in (23) to solve the expectation-based form of the stochastic optimization problem given by (10).

It could be argued that a robust policy should be used to solve a robust problem. Just as (22) is a static robust problem, we might pose a sequential robust problem as

$$\min_{\pi \in \Pi} \max_{\omega \in \mathcal{W}} \sum_{t=0}^{T} C\Big(S_t(\omega), X_t^{\pi}\big(S_t(\omega)\big)\Big). \tag{24}$$

Finding the optimal policy that solves (24) appears to be computationally intractable. It is harder than (10) since the expectation is an additive operator, while $\max_{\omega \in \mathcal{W}}$ is not additive.

4 Closing remarks

It is not unusual to find papers simulating two policies, averaging the results and then reporting which one is best. However, it is surprisingly difficult to find papers that will actually write out this function as we have in equation (12), and even rarer to see this expressed as an approximation of the expectation in equation (10). However, if a paper compares two policies by averaging the results of simulations, as is done in [2], it seems unavoidable that the authors should recognize that their true objective function is, in fact, (10), and they are doing an extremely limited search for the optimal policy.

We argue that any paper that is testing and comparing different policies for a sequential optimization problem should start by writing (10) as the objective, just as we write the objective for a deterministic optimization problem as is illustrated in equation (1). In the process, we must acknowledge we are searching for the best policy, just as we searched over vectors $x \in \mathcal{X}$. We need to acknowledge the full range of policies that might be tested, and even if we cannot test them all, perhaps we can replace our current culture of rooting for a policy class like our favorite sports team, and simply let scientific experimentation decide this problem for us.

A series of tutorials on this topic are available at www.castlelab.princeton.edu/jungle.htm.

Warren B. Powell, Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ, USA. powell@princeton.edu

Bibliography

[1] Ben-Tal, A., Ghaoui, L. E. & Nemirovski, A. (2009), *Robust Optimization*, Princeton University Press, Princeton NJ.

[2] Ben-Tal, A., Golany, B., Nemirovski, A. and Vial, J.-P. (2005), 'Retailer-Supplier Flexible Commitments Contracts: A Robust Optimization Approach', *Manufacturing and Service Operations Management* **7**(3), 248–271.

[3] Birge, J. R. & Louveaux, F. (2011), *Introduction to Stochastic Programming*, 2nd edn, Springer, New York.

[4] Hastie, T., Tibshirani, R. and Friedman, J. (2009), *The elements of statistical learning: data mining, inference and prediction*, Springer, New York.

[5] Powell, W. B. (2011), *Approximate Dynamic Programming: Solving the curses of dimensionality*, 2 edn, John Wiley & Sons, Hoboken, NJ.

[6] Shapiro, A., Dentcheva, D. & Ruszczyński, A. (2014), *Lectures on stochastic programming: modeling and theory*, 2 edn, SIAM, Philadelphia.

[7] Rockafellar, R. T. & Wets, R. J.-B. (1991), 'Scenarios and policy aggregation in optimization under uncertainty', *Mathematics of Operations Research* **16**(1), 119–147.

Andrzej Ruszczyński

Few Remarks on Stochastic Dynamic Optimization

Warren Powell provides a great service to our community with his article. Indeed, many people in many areas work on similar problems under different names and come to similar conclusions.

His view may be appealing, because it attempts to reduce the complexity of several research areas to one problem formulation and several basic ideas of solution methods. As usually happens in such cases, the most difficult are the first steps, and the foundations of his proposal need to be carefully examined. While this commentary is not the best platform to carry out such analysis, I will restrict my remarks to four issues which caught my attention and which need to be examined with more scrutiny.

1 Wealth of Problem Formulations

An important feature of dynamic stochastic optimization, which has been overlooked in the article, is the wealth of problem formulations in this area. While deterministic optimization theory focuses on a small number of well-defined problems, optimization under uncertainty introduces a great variety of problem formulations, involving expected value optimization, probabilistic constraints, stochastic

dominance relations, risk measures, reliability or bankruptcy considerations, etc. In my opinion, the restriction of possible problem formulations to just one or few possibilities may be very detrimental to the development of the area. In fact, new problem formulations are sources of qualitative progress of the field.

2 Stochastic Programming versus Dynamic Programming

Having our earlier remarks in mind, we may still admit that model (10), which is used by Dr. Powell as the base model of multistage stochastic optimization, does indeed coincide with many popular problem formulations. However, superficially similar models may serve different purposes for different applications. The main difference between stochastic programming and dynamic programming models is, in my opinion, the relative importance of the first-stage decision.

In stochastic programming, models of form (10) are usually constructed to find the first-stage decision that allows for efficient response to uncertain circumstances which may occur in the future. The nature of the first-stage decision x_0 is usually different than later decisions. A good example is a stochastic facility location problem, where the first-stage decision is to choose locations of several service units, while the second-stage and later decisions represent the service provided by these facilities to respond to uncertain demand. The problem is mainly about locating the facilities, and the later stages serve as a model of uncertain future operation conditions. The horizon T can be chosen to represent these conditions in the best possible way. While it is possible to consider location of facilities over two or three periods, it is hard to envisage an infinite-horizon problem of this type.

A mathematical advantage of the focus on a few earlier stages is the possibility to exploit convexity, which helps analyze and solve stochastic programming problems of significant size. Furthermore, these models allow for non-Markovian structure of the underlying process.

In dynamic programming, on the other hand, the focus is on the long-term performance of the system. The goal is to find a decision rule (feedback control) to ensure good operation of the system in the long run. The decisions made at successive periods are of the same nature; frequently, we make stationarity assumptions about the dynamics of the system and external disturbances, and every stage looks like the first stage. That is why we can successfully model such problems as infinite-horizon dynamic programming problems. The extreme case is the average cost problem, in which the first stage does not matter much. Dr. Powell is right in observing that stochastic programming models may be used as tools for deriving approximate look-ahead policies in dynamic programming, but the nature of both classes of models is different.

3 Stochastic versus Robust Optimization

I share Dr. Powell's criticism of robust optimization models, but I think that his brief discussion of these approaches missed one very important issue. While the technical property of additivity is important, what matters most are the ways of modeling dependence. In a system affected by several uncertain quantities, relations between these uncertain quantities are germane for the operation of the system and for developing a policy. In dynamical systems, in which we are interested in the long-term effects of our decisions, we must not ignore relations between future uncertain outcomes and past observations. The theory of probability and stochastic processes provides us with powerful tools to model dependence and to use it for guiding our actions. I am not aware of the existence of such tools in robust optimization.

A closely related issue is the construction of uncertainty sets, which are the key ingredients of robust optimization problems. While in a static setting one might envisage some confidence sets playing this role, the dynamic situation is completely unclear. Should the confidence sets be formulated for the entire sequence of random outcomes? How do they change, once the earlier outcomes have been observed? Do policy recommendations enjoy the time-consistency property, or is it possible that we shall have an incentive to renege on our plans, once some observations have been collected? The theory of robust optimization has not yet provided convincing answers to these questions.

4 Expected Value versus Risk Measures

Because of the lack of space, Dr. Powell glossed over one of the most fascinating recent development in stochastic dynamic optimization: the use of dynamic measures of risk. There is far more to it than just replacing the expected value operator $\mathbb{E}$ in (10) by ϱ in (11). The use of dynamic measures of risk creates a large number of theoretical and computational difficulties: we need to ensure the time-consistency property of the model, we need to understand what Markov property means in this setting, we need efficient solution methods, and we need new statistical techniques, far beyond the sample-based model (12).

I want to finish my remarks with the statement that satisfactory answers to the issues raised in Dr. Powell's paper and in my short note cannot be found without mathematics. Only solid mathematical foundations may lead to a theory that can stand the test of time and become practically useful.

Andrzej Ruszczyński, Department of Management Science and Information Systems, Rutgers University, Piscataway, New Jersey, USA.
rusz@business.rutgers.edu

Warren B. Powell

Response to Andrzej Ruszczynski's comments

Andrzej has made a number of insightful comments about the article and the field. There is one issue, however, that needs a response. He makes the comment "In stochastic programming, models of form (10) are usually constructed to find the first-stage decision that allows for efficient response to uncertain circumstances which may occur in the future." It is certainly true that a large part of the stochastic programming literature focuses on two-stage *problems*, but this article specifically addresses modeling issues associated with sequential problems. There are many applications of stochastic programming in operational problems in areas such as transportation, logistics and energy. The stochastic unit commitment problem (which involves the planning of when energy generators should be turned on or off) is a particularly popular area of application. These are fully sequential problems where decisions have to be made over time.

Andrzej acknowledges that there are some unresolved modeling issues in the robust optimization community (in the setting of fully sequential problems), but his response does not recognize that the same issues arise in the use of stochastic programming (again, only in the setting of sequential problems). Stochastic programs, when used

for sequential problems, are a form of lookahead policy that represents one possible solution to equation (10). It is virtually automatic for an author applying stochastic programming to a sequential problem to write a model in the form of (21), but not using (10).

The point of this article is to make the case that equation (10) is the true objective function, and should be written explicitly. Then, the stochastic program in equations (21) is simply one class of policy. Authors should recognize that there are competing classes (e.g. deterministic lookaheads, robust cost function approximations or even robust lookahead models), in addition to testing variations within a class (e.g. variations of the number of scenarios, and possibly comparisons of two-stage and three-stage approximations). Equation (10), then, becomes the basis for deciding which policy is best.

ISMP 2015 in Pittsburgh: Plenary and Semi-Plenary Speakers

The 22nd *International Symposium on Mathematical Programming* (ISMP 2015) will take place in Pittsburgh, PA, USA, July 12–19, 2015. ISMP is a scientific meeting held every three years on behalf of the *Mathematical Optimization Society* (MOS). It is the world congress of mathematical optimization where scientists as well as industrial users of mathematical optimization meet in order to present the most recent developments and results and to discuss new challenges from theory and practice.

Plenary speakers

Laurent El Ghaoui (University of California, Berkeley). Laurent El Ghaoui graduated from Ecole Polytechnique (Palaiseau, France) in 1985, and obtained his Ph.D. in Aeronautics and Astronautics at Stanford University in 1990. He taught at several institutions in France, including Ecole Polytechnique, before joining the EECS department at UC Berkeley in 1999. His research interests include robust optimization, large-scale machine learning, with a focus on text analytics.

Jim Geelen (University of Waterloo, Canada). Jim Geelen completed his Ph.D. at the University of Waterloo. After three postdoctoral fellowships at CWI (Amsterdam), RIMS (Kyoto), and ZPR (Cologne), he returned to the University of Waterloo. For the past 15 years he, together with Bert Gerards and Geoff Whittle, has been working on extending the graph minors project to binary matroids.

Daniel Kuhn (EPFL, Switzerland). Daniel Kuhn holds the Chair of Risk Analytics and Optimization at EPFL. Before joining EPFL, he was a faculty member at Imperial College London (2007–2013) and a postdoctoral researcher at Stanford University (2005–2006). He received a Ph.D. in Economics from the University of St. Gallen in 2004 and an M.Sc. in Theoretical Physics from ETH Zürich in 1999. His research interests revolve around robust optimization and stochastic programming.

Daniel A. Spielman (Yale University). Daniel Alan Spielman received a B.A. in Mathematics and Computer Science from Yale in 1992 and a Ph.D. in Applied Mathematics from M.I.T. in 1995. He spent a year as a NSF Mathematical Sciences Postdoc in the Computer Science Department at U.C. Berkeley, and then taught in the Applied Mathematics Department at M.I.T. until 2005. Since 2006, he has been a Professor of Computer Science and Mathematics at Yale University. He has received many awards, including the 1995 ACM Doctoral Dissertation Award, the 2002 IEEE Information Theory Paper Award, the 2008 Godel Prize, the 2009 Fulkerson Prize, the 2010 Nevanlinna Prize, the 2014 Polya Prize, an inaugural Simons Investigator Award, and a MacArthur Fellowship. His main research interests include the design and analysis of algorithms, network science, machine learning, digital communications and scientific computing.

Stephen J. Wright (University of Wisconsin). Stephen Wright received a B.Sc.(Hons) degree in 1981 and a Ph.D. in 1984 from the University of Queensland. He has held appointments at the University of Arizona, North Carolina State University, Argonne National Laboratory (during the 1990s), and the University of Chicago. Since 2001 he has been at the University of Wisconsin-Madison. His research is in continuous optimization and its applications to all areas of science and engineering.

Semi-plenary speakers

Samuel A. Burer (University of Iowa). Sam Burer is Professor and Tippie Research Fellow in the Department of Management Sciences at the University of Iowa. He received his Ph.D. from the Georgia Institute of Technology, and his research and teaching interests include convex optimization, mixed integer nonlinear programming, operations research, and management sciences. His research has been supported by grants from the National Science Foundation, and he serves on the editorial board of Operations Research, SIAM Journal on Optimization, Mathematics of Operations Research, and Optima. He also serves as a Council Member of the Mathematical Optimization Society, and as a Member of the Board of Directors of the INFORMS Computing Society.

Roberto Cominetti (University of Chile, Chile). Roberto Cominetti graduated as Mathematical Engineer from Universidad de Chile in 1986 and received a Ph.D. in Applied Mathematics from Université Blaise Pascal (Clermontt II) in 1989. He has developed his career at the University of Chile, first at the Department of Mathematical Engineering and more recently at the Department of Industrial Engineering. His main research interests are in convex optimization and algorithmic game theory as well as their applications to equilibrium and dynamics in transportation networks.

Michelangelo Conforti (University of Padova, Italy). Michele Conforti received his BS from University of Bologna and a Ph.D. from Carnegie Mellon University. He is currently professor in the Mathematics Department, University of Padova. His interests are mainly in Combinatorial Optimization and Graph Theory. He is co-recipient of the Fulkerson Prize. In the past years he has worked in Integer Programming and has recently co-authored a book on the subject.

Tamara G. Kolda (Sandia Labs). Tamara Kolda is a Distinguished Member of the Technical Staff at Sandia National Laboratories in Livermore, California. Her research interests include multilinear algebra and tensor decompositions, graph models and algorithms, data mining, optimization, nonlinear solvers, parallel computing and the design of scientific software. She received her Ph.D. from the University of Maryland in 1997 and was the Oak Ridge National Lab Householder Postdoc in Scientific Computing from 1997-99.

Andrea Lodi (University of Bologna, Italy). Andrea Lodi received the Ph.D. in System Engineering from the University of Bologna in

2000 and he has been Herman Goldstine Fellow at the IBM Mathematical Sciences Department, NY in 2005–2006. He is full professor of Operations Research at DEI, University of Bologna since 2007. His main research interests are in Mixed-Integer Linear and Nonlinear Programming. He serves as Associated Editor for several prestigious journals in the area and is currently network coordinator of the two EU projects, and, since 2006, consultant of the IBM CPLEX research and development team.

Asu Ozdaglar (Massachusetts Institute of Technology). Asu Ozdaglar received the B.S. degree in electrical engineering from the Middle East Technical University, Turkey, in 1996, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology in 2003. She is currently a professor in the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology. She is the director of the Laboratory for Information and Decision Systems and also a member of the Operations Research Center. Her research area includes optimization theory, with emphasis on nonlinear programming, distributed optimization, and convex analysis, game theory, with applications in communication, social, and economic networks, and network analysis with special emphasis on contagious processes, systemic risk and dynamic control.

Werner Römisch (Humboldt University Berlin, Germany). Werner Römisch received a mathematics diploma in 1971 and a Ph.D. in 1976 both from Humboldt-University Berlin. He continued at Humboldt University Berlin and received there a habilitation in 1985 and a full professorship in 1993. His research is mainly in stochastic optimization with side interests in stochastic equations and risk, and applications in energy and revenue management.

Frank Vallentin (University of Köln, Germany). Frank Vallentin is a professor of applied mathematics and computer science at Universität zu Köln. In 2003 he received his Ph.D. in mathematics from Technische Universität München. Past appointments include assistant and associate professor at Technische Universiteit Delft, postdoc at Centrum Wiskunde & Informatica in Amsterdam and postdoc at the Hebrew University of Jerusalem. His research interests include optimization, geometry, discrete and experimental mathematics.

Pascal van Hentenryck (NICTA, Australia). Pascal Van Hentenryck got his B.Sc. and Ph.D. from the University of Namur in Belgium. He leads the Optimization Research Group at NICTA and holds a vice-chancellor chair in data-intensive computing at the Australian National University. Prior to his NICTA appointment, he was professor at Brown University for about 20 years. His current research is at the intersection of data science and optimization with applications in energy, logistics, disaster management, and computational social science.

Ya-xian Yuan (Chinese Academy of Sciences, China). Ya-xiang Yuan received a B.Sc. from Xiangtan University (China) in 1981 and a Ph.D. from University of Cambridge (UK) in 1986. He was Rutherford Fellow at Fitzwilliam College, University of Cambridge from 1985–1988. He returned to China in 1988, and has been working as a full professor at the Chinese Academy of Sciences since then. His research area is mainly in continuous optimization, particularly on trust region methods, quasi-Newton methods, nonlinear conjugate gradients, and subspace methods.

Information

ISMP details (including clusters, cluster chairs, important dates, all registration rates, discounted MOS membership for 2016–2018, hotel, sponsorship opportunities, exhibits etc.) are available on the conference web site at www.ismp2015.org.

Application for Membership

I wish to enroll as a member of the Society. My subscription is for my personal use and not for the benefit of any library or institution.

☐ I will pay my membership dues on receipt of your invoice.
☐ I wish to pay by credit card (Master/Euro or Visa).

Credit card no. *Expiration date*

Family name

Mailing address

Telephone no. *Telefax no.*

E-mail

Signature

Mail to:
Mathematical Optimization Society
3600 Market St, 6th Floor
Philadelphia, PA 19104-2688
USA

Cheques or money orders should be made payable to The Mathematical Optimization Society, Inc. Dues for 2014, including subscription to the journal *Mathematical Programming*, are US $90. Retired are $45. Student applications: Dues are $22.50. Have a faculty member verify your student status and send application with dues to above address.

Faculty verifying status

Institution

IMPRINT
Editor: Volker Kaibel, Institut für Mathematische Optimierung, Otto-von-Guericke Universität Magdeburg, Universitätsplatz 2, 39108 Magdeburg, Germany. kaibel@ovgu.de ■ **Co-Editors:** Samuel Burer, Department of Management Sciences, The University of Iowa, Iowa City, IA 52242-1994, USA. samuel-burer@uiowa.edu ■ Jeff Linderoth, Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI 53706-1572, USA. linderoth@wisc.edu ■ **Founding Editor:** Donald W. Hearn ■ Published by the Mathematical Optimization Society ■ Design and typesetting by Christoph Eyrich, Berlin, Germany. optima@0x45.de ■ Printed by Oktoberdruck AG, Berlin, Germany.