INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Parallel Nonstationary Direct Policy Search for Risk-Averse Stochastic Optimization

Somayeh Moazeni, Warren B. Powell, Boris Defourny, Belgacem Bouzaiene-Ayari

To cite this article:
Somayeh Moazeni, Warren B. Powell, Boris Defourny, Belgacem Bouzaiene-Ayari (2017) Parallel Nonstationary Direct Policy Search for Risk-Averse Stochastic Optimization. INFORMS Journal on Computing 29(2):332-349. http://dx.doi.org/10.1287/ijoc.2016.0733

http://pubsonline.informs.org/journal/ijoc/

INFORMS JOURNAL ON COMPUTING
Vol. 29, No. 2, Spring 2017, pp. 332–349
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

Parallel Nonstationary Direct Policy Search for Risk-Averse Stochastic Optimization

Somayeh Moazeni,[a] Warren B. Powell,[b] Boris Defourny,[c] Belgacem Bouzaiene-Ayari[b]

[a] School of Systems and Enterprises, Stevens Institute of Technology, Hoboken, New Jersey 07030; [b] Department of Operations Research and Financial Engineering, Princeton University, Princeton, New Jersey 08544; [c] Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, Pennsylvania 18015

Contact: smoazeni@stevens.edu (SM); powell@princeton.edu (WBP); defourny@lehigh.edu (BD); belgacem@princeton.edu (BB-A)

Received: March 16, 2015
Revised: January 11, 2016; July 1, 2016; September 11, 2016
Accepted: September 13, 2016
Published Online: April 12, 2017

http://dx:doi.org/10.1287/ijoc.2016.0733

Abstract. This paper presents an algorithmic strategy to nonstationary policy search for finite-horizon, discrete-time Markovian decision problems with large state spaces, constrained action sets, and a risk-sensitive optimality criterion. The methodology relies on modeling time-variant policy parameters by a nonparametric response surface model for an indirect parametrized policy motivated by Bellman's equation. The policy structure is heuristic when the optimization of the risk-sensitive criterion does not admit a dynamic programming reformulation. Through the interpolating approximation, the level of nonstationarity of the policy, and consequently, the size of the resulting search problem can be adjusted. The computational tractability and the generality of the approach follow from a nested parallel implementation of derivative-free optimization in conjunction with Monte Carlo simulation. We demonstrate the efficiency of the approach on an optimal energy storage charging problem, and illustrate the effect of the risk functional on the improvement achieved by allowing a higher complexity in time variation for the policy.

History: Accepted by Alice Smith, Area Editor for Heuristic Search.
Funding: This research was supported, in part, by a grant from the Lawrence Livermore National Laboratory and a postdoctoral fellowship from Natural Sciences and Engineering Research Council of Canada [Grant NSERC-PDF-438978-2013].
Supplemental Material: The online supplement is available at https://doi.org/10.1287/ijoc.2016.0733.

Keywords: dynamic optimization • risk-averse stochastic optimization • parallel optimization • derivative-free optimization • direct policy search • learning • energy storage

1. Introduction

This paper addresses computational challenges arising in the direct policy search approach to approximate dynamic programming (ADP) for problems where the optimal policy is known to be nonstationary. ADP allows the application of dynamic programming to large-scale Markov decision processes (MDPs) by using approximations and simulation (Bertsekas and Tsitsiklis 1996, Sutton and Barto 1998, Powell 2011, Bertsekas 2012).

Direct policy search broadly refers to methods in ADP where the policy is represented by a function approximator, including some tuning parameters to be optimized. Direct policy search methods differ by the approximation architecture and by the stochastic optimization method used to tune the policy parameters. For nonstationary policies, which can change at each decision stage, the dimension of this embedded stochastic search problem grows linearly with the time horizon (see, e.g., Powell 2011). The high dimensionality of the parameter space complicates the optimization of the approximate policy tremendously. In addition, depending on the policy function approximation architecture, the stochastic search problem typically becomes a nonconvex or semi-infinite optimization problem.

To address these challenges, we study direct policy search with a nonstationary policy that minimizes a parametric cost function approximation. We let the time-dependent parameters vary according to an interpolating response surface model. More precisely, we assume that the policy is parameterized by a relatively low dimensional vector θ at each point in time t. Each element $\theta_i(t)$ can then be modeled using an interpolating response surface model. In our work, we use spline functions, which have been demonstrated to be a versatile approximation architecture for solving dynamic programs (Johnson et al. 1993, Chen et al. 1999). The nonstationarity of the policy can then be adjusted through the number of interpolating knots over the time horizon. This approximation considerably reduces the size of the search problem and enables the model user to actively control the dimensionality and to balance the trade-off between accuracy and computational cost.

The cost function approximation-based policy, which we describe in detail later on, readily ensures that for a given state and set of tuning parameters, the computed action is feasible. This feature makes the direct policy search framework amenable to handling a large set of constraints on the actions. In addition, computing optimal feasible actions associated with a set of states and policy parameters can be parallelized.

We use the fact that any orderpreserving transformation of the value function can yield optimal decisions. Hence, a near-optimal policy does not necessarily rely on a near-optimal representation of the value function. Furthermore, when an appropriate architecture is applied in the policy function approximator, this policy parametrization is compatible with an exact dynamic programming solution, assuming that the minimization of the optimality criterion admits a dynamic programming reformulation. Otherwise, the dynamic programming-inspired policy structure could be viewed as a heuristic to minimize a general risk-sensitive objective.

The policy parameters are often tuned to optimize a real-valued function of the costs. Examples of criteria for MDPs include the expected value or a given risk measure of the total cumulated cost over the planning horizon. In this paper, we do not make any assumption on the structure (e.g., separability, convexity, or differentiability) of this optimality criterion. In addition, due to the minimization operator in the policy function approximation, the search for a set of near-optimal tuning parameters becomes less structured and has a nonconvex feasible space, even when state spaces are convex and compact. To keep the generality of the approach to handle any optimality criterion, we resort to derivative-free optimization approaches, which admit a parallel implementation, see, e.g., Bertsekas and Tsitsiklis (1989), Dixon and Jha (1993), Golub and Ortega (1993), Yamakawa and Fukushima (1996), Censor and Zenios (1998), and Rios and Sahinidis (2013). In this paper, we adopt a multistart pattern search framework (Audet and Dennis 2002) combined with Monte Carlo simulation.

The use of derivative-free search on the interpolating knots of the parameter vector of the cost function approximation enables us to massively parallelize the direct policy search and take advantage of high performance computing. Our parallel implementation includes three layers of parallelization; namely, for the starting points, optimization iterations, and the objective function evaluations. The three levels of parallelization significantly improve the efficiency and speed of the algorithm implemented on a multicluster supercomputer, as demonstrated by the numerical experiments reported in this paper. Our parallel programming model is a hybrid model of message-passing model (distributed parallel computing) and multithreading. The multithreaded component of the program allows us to adopt the framework and get some speed up on computer systems with multicore processors.

We investigate the efficiency and tractability of the approach by using an example from energy storage operation management. We study optimal charging policies for a storage device in the presence of a nonstationary stochastic electricity price process, nonstationary stochastic load evolution, and an intermittent renewable energy resource. The goal is to find an optimal storage charging policy over a finite-time horizon, which minimizes some risk functional of the total system operations cost, while fully serving the electricity load.

We first formulated the problem with a stylized storage model, in which the device is fast ramping and fully efficient. As a contribution of independent interest, we provide a solution for the exact risk-neutral stochastic dynamic programming policy and its corresponding optimal value function. We use this closed-form solution as a reference solution to benchmark our approach. In particular, we show that by using the proposed direct policy search strategy, the policy converges to an exact optimal solution to the dynamic program. Next, we investigate the proposed computational stochastic optimization framework to minimize the value at visk (VaR), conditional value at risk (CVaR), and expectation of the cost. The VaR is a widely used and standard risk measure in various fields, such as finance (Duffie and Pan 1997) or energy (Burger et al 2008), but it is a nonconvex, nonsmooth risk measure (Rockafellar and Uryasev 2000). We observe that the impact of the nonstationarity assumption depends on the risk functional. Increasing the level of nonstationarity can significantly improve the performance of the computed charging policy under the VaR.

The proposed approach is implemented on the *Hyperion* cluster from the Lawrence Livermore National Laboratory and the *Tower* cluster of CASTLE Labs at Princeton University. The impact of using multiple processors on the runtime of the developed methodology is reported.

The main research contribution of our paper is to address computational challenges in direct policy search when the policy is parameterized indirectly using a minimization operator, the policy parameters are time dependent, the state space is high dimensional, the actions are constrained, and a risk measure is used in the objective. In our direct policy search approach, there is no attempt to estimate the value function.

The presentation is organized as follows. We review the relevant literature in Section 2. In Section 3, we present the basic components of MDPs and formalize

the concept of level of nonstationarity for nonstationary direct policy search. The proposed cost function approximation policy and the proposed nested hybrid massively parallel implementation of the approach are also described in this section. The motivating application in the context of energy storage operations management is discussed in Section 4. The results of the numerical study on computational tractability and algorithmic performance are reported in Section 5. We conclude the paper in Section 6.

2. Literature Review

Our work is related to several areas of research, as we now describe. Direct policy search methods have been the subject of several studies in the machine learning and operations research communities (Busoniu et al. 2010, Giuliani et al. 2015). When the gradient of the policy function approximation with respect to the policy parameters can be estimated, a class of techniques, called policy-gradient methods, performs gradient descent to find local optima in the parameter space. For applications of policy-gradient methods in direct policy search, see, e.g., Sutton et al. (2000), Baxter and Bartlett (2001), Kakade (2002), Peters et al. (2005), Kober and Peters (2009), Silver et al. (2014), and the references therein. Policy-gradient methods assume that the policy is differentiable in its parameters. In addition, they are often very slow and require a large number of iterations to converge, when the dimension of parameter space increases, which is the case in nonstationary policy search. Ng and Jordan (2000) apply exhaustive enumeration to find the best policy for a finite set of actions. For a sigmoidal policy parameterization, they adopt a gradient descent to optimize over the policy parameters. Strens and Moore (2003) investigate the use of paired statistical tests for comparing policies, aiming to find a globally optimal policy, without estimating a gradient. They illustrate the approach when the parameterized policy is given by a thresholded polynomial in the state with two parameters. Mannor et al. (2003) use the cross entropy method for optimization in the policy space, assuming that the policy is parameterized by a small number of parameters, e.g., a threshold policy. Kormushev and Caldwell (2012) propose an algorithm based on particle filtering to perform global search in policy parameter space. They validate the technique on one- and two-dimensional problems. Actor-only methods in actor-critic reinforcement learning, surveyed in Grondman et al. (2012), also work with a parameterized family of policies and optimize the cost directly over the policy parameter space. This class of methods uses an indirect parametrization that has similarities with the policy representation studied in the present paper; however, the present work differs in that the updates in actor-critic methods are generally performed based on a policy-gradient formula, and that the policy optimizes a state-action value function approximation. The literature on computational aspects or optimality of parametric policies has been mainly concerned with affine policies, see, e.g., Bertsimas et al. (2010), Kuhn et al. (2011), Bertsimas and Goyal (2012), and Moazeni et al. (2016). For stationary policy search with a small number of parameters, Scott et al. (2014) use the knowledge gradient for continuous parameters policy (Scott et al. 2011) to find policy parameters.

Variations of the optimal storage charging management problem have been studied in the literature. The sizing and management of a hydro pumping storage unit in an island power system with renewable penetration are studied in Brown et al. (2008). The objective is to minimize the expected operation and annualized installation costs over all scenarios of wind, hydropower production, and load. Gonzalez et al. (2008) discuss the maximization of the profit of a wind farm and a hydro pumped storage unit owner over a finite horizon, in compliance with commitments in the market. The problem is formulated as a two-stage stochastic programming problem. The problem of maximizing the total discounted revenue of a storage facility owner, assuming an exogenous stochastic process for the electricity price, is investigated in Carmona and Ludkovski (2010), and a simulation-based numerical method is proposed. Lai et al. (2010) study the problem of the real option valuation for natural gas storage, in the presence of capacity constraints and uncertainty in the natural gas price dynamics. They develop a tractable ADP approach, coupled with Monte Carlo simulation, to compute bounds on the storage value. An optimal energy commitment policy for the owner of a wind farm and storage resources to maximize the revenue in an infinite-time horizon is studied in Kim and Powell (2011), where electricity price and wind power supply evolve as stochastic processes. Harsha and Dahleh (2015) consider a setting that involves conventional generation with a real-time price, renewable generation with a zero marginal cost, and energy storage. The goal is to minimize the cost of generation and investment in storage, while fully serving the demand. The authors formulate the problem as a discrete-time stochastic dynamic programming problem over an infinite horizon and establish the existence of an optimal stationary policy. Zhou et al. (2014) consider the colocation of wind generation and electricity storage facilities with uncertain wind and stochastic prices, limited transmission capacity, and the option of buying power from the market. They model the wind-storage management problem as a finite-horizon MDP and establish some structural results. Moazeni et al. (2015) discuss the importance of downside risk considerations in managing energy storage systems exposed to price risk, due to the fat-tailed distribution of energy

prices. Scott et al. (2014) show that, in the context of a storage problem, direct policy search, even for stationary policies, can outperform several approximate policy iteration algorithms. In this paper, we adopt a model similar to the ones considered in Moazeni et al. (2015) and Scott et al. (2014).

3. Nonstationary Direct Policy Search and Risk Aversion

We consider a discrete-time dynamic optimization problem described by the following components:

- *States*: The state S_t, valued in a subset $\mathcal{S}_t$ of a state-space $\mathcal{S}$.
- *Decisions*: The vector-valued decision x_t, to be in the feasible set $\mathcal{X}_t(S_t)$. The notation stresses that $\mathcal{X}_t(S_t)$ can depend on the state S_t and time t.
- *Exogenous information*: The random vector W_{t+1}, assumed to be independent of S_t and x_t. The distribution of W_{t+1} depends on the period index t.
- *Transition function*: The next state is described by the model $S_{t+1} = S^M_{t+1}(S_t, x_t, W_{t+1})$. The time index in S^M_{t+1} stresses that the transition function is nonstationary.
- *Cost function*: The cost function is described by $C_t(S_t, x_t)$. The time index in C_t stresses that the cost function is nonstationary.

The dynamic optimization problem under consideration is written as

$$\text{minimize } J^\pi(s) \stackrel{\text{def}}{=} \rho\left(\sum_{t=0}^{T-1} C_t(S_t, X^\pi_t(S_t)) \;\middle|\; S_0 = s\right). \quad (1)$$

The optimization is over a class Π of nonstationary policies $\pi = (\pi_0, \ldots, \pi_{T-1})$. The policy π is said to be stationary if π_t is independent of t. We write $X^\pi_t(S_t)$ to indicate that the decision X_t is a function of the state S_t determined by the policy π at time step t. The optimal value of problem (1) is denoted by $J(s) \stackrel{\text{def}}{=} \min_{\pi\in\Pi} J^\pi(s)$.

The objective function is expressed with the risk functional ρ, which maps the random cost

$$B^\pi(S_0) \stackrel{\text{def}}{=} \sum_{t=0}^{T-1} C_t(S_t, X^\pi_t(S_t)) \quad (2)$$

to a real number. We write B^π and J^π to stress that the distribution of the random return depends on the policy π.

In the literature on MDPs, often the expectation $\mathbb{E}$ plays the role of the risk functional ρ. Alternative examples of risk functionals include the variance, the semivariance, the VaR, and the CVaR. Positive weighted sums of risk functionals also define risk functionals. It is well known that the MDPs with certain risk functionals, such as the VaR, is not amenable to stochastic dynamic programming techniques and an appropriate recursive optimality equation using value functions cannot be established, as it can be done with the expectation, see, e.g., Shapiro (2012) or the examples given in Defourny et al. (2008) and Rudloff et al. (2014).

In the present work, we therefore focus on direct policy search methods, which seek to find a best policy within a class of parameterized policies, by directly optimizing the performance measure ρ over the time-varying policy parameters simultaneously. To deal with the computational challenges posed by the large number of policy parameters and constraints on the decisions in direct policy search, we consider a class of policies, which captures the effect of the risk-adjusted "value" of the distribution of the next state, conditioned on the current state, by an approximation architecture where the time-varying policy parameters follow an interpolating response surface model over time. These steps are explained in the following sections.

3.1. Cost Function Approximation Policy

We carry out policy search over a class of nonstationary deterministic policies given by

$$X^\pi_t(s_t \mid \theta) \in \arg\min_{x_t\in\mathcal{X}_t(s_t)} \{C_t(s_t, x_t) + \mathbb{K}_{\theta_t}(S^M_{t+1}(s_t, x_t, W_{t+1}))\}, \quad t = 0, \ldots, T-1, \quad (3)$$

where $\mathbb{K}_{\theta_t}$ gives a risk-adjusted value of the distribution of the next state $S_{t+1} = S^M_{t+1}(s_t, x_t, W_{t+1})$ given the current state s_t and decision x_t. The risk-adjusted value depends on the parameter vector θ_t. When the risk functional is the expectation, $\mathbb{K}_{\theta_t}(S^M_t(s_t, x_t, W_{t+1}))$ plays the role of the expected value function at the next state. We refer to

$$\mathcal{H}_{\theta_t}(s_t, x_t) \stackrel{\text{def}}{=} \mathbb{K}_{\theta_t}(S^M_{t+1}(s_t, x_t, W_{t+1})) \quad (4)$$

as the *correction function*.

When $\mathcal{H}_{\theta_t}(s_t, x_t) \equiv 0$ is chosen in (3) for $t = 0, \ldots, T-1$, $X^\pi_t(s_t \mid \theta)$ is reduced to the myopic policy, i.e., the decision maker has simply chosen to ignore the fact that decisions at every time step may impact the future system state and total cost. Therefore the function $\mathcal{H}_{\theta_t}(s_t, x_t)$ can be interpreted as a correction to adjust the myopic policy to capture the effect of future costs.

We assume that the minimizations in (3) are attained. This assumption holds, for example, when $\mathcal{X}_t(s_t)$ is a compact set and $C_t(s_t, x_t) + \mathcal{H}_{\theta_t}(s_t, x_t)$ is a continuous function of x.

A natural choice to represent the correction function $\mathcal{H}_{\theta_t}$ is a linear approximation architecture. This model is described by a linear combination of basis functions taken from a finite collection $\{\phi_k(s, x)\}_{k=1}^K$ of functions, assumed to be specified beforehand and easily computable,

$$\mathcal{H}_{\theta_t}(s_t, x_t) = \sum_{k=1}^K \theta_{t,k}\, \phi_k(s_t, x_t). \quad (5)$$

It turns out that only basis functions with $\nabla_x \phi_k(s_t, x_t) \neq 0$ may influence the computation of $X_t^\pi(s_t \mid \theta)$. This is because (i) minimizers are insensitive to adding a constant and (ii) shifting the correction function relative to different states or times has no impact on other correction functions since correction functions are not computed recursively. We note that the linear approximation architecture is not the only choice. The optimization approach developed in this paper can accommodate other types of approximation, as long as the minimizations in (3) remain computationally tractable.

The cost function approximation in Equation (3) using the linear approximation architecture (5) is then fully defined by the $K \times (T-1)$ parameters $\theta_{t,k}$, which will govern (along with the initial choice of basis functions) the performance of the policy. We assume that $\theta \stackrel{\text{def}}{=} (\theta_0, \ldots, \theta_{T-1})$ can be valued in a compact set $\Theta \stackrel{\text{def}}{=} \Theta_0 \times \cdots \times \Theta_{T-1}$, and describe the optimization as

$$\min_{\theta_0,\ldots,\theta_{T-1}} \quad \rho\left(\sum_{t=0}^{T-1} C_t(S_t, X_t^\pi(S_t \mid \theta)) \,\middle|\, S_0 = s\right) \qquad (6)$$
$$\text{s.t.} \quad \theta_t \in \Theta_t, \quad t = 0, \ldots, T-1.$$

The optimal value of problem (6) is denoted by $\hat{J}(s) \stackrel{\text{def}}{=} \min_{\pi \in \mathcal{H}^\pi} J^\pi(s)$, where $\mathcal{H}^\pi$ denotes the space of policies expressed as in (3). A sufficient condition for $\hat{J}(s) = J(s)$ is $\mathcal{H}^\pi = \Pi$. Recall that $J(s)$ is the optimal value of problem (1). Problem (6), along with the parametric cost function approximation in Equation (3), aims to heuristically solve problem (1).

As an interesting feature of the policy parametrization in (3), the argmin operator takes into account the constraints defining admissible decisions, and thereby ensures that the computed decision is feasible. Therefore, problem (6) only includes constraints in Θ_t, e.g., only box constraints.

Alternative constraints on policy parameters may also be included in problem (6). For instance, we can impose that the policy be stationary up to the time step $T-2$ and be myopic at $T-1$ through the additional constraints $\theta_0 = \cdots = \theta_{T-2}$ and $\theta_{T-1} = 0$.

Appropriate assumptions on the correction function $\mathcal{K}_{\theta_t}$ and the parameter sets Θ_t ensure that an exact stochastic dynamic programming policy is included as a feasible policy in problem (6). This is more precisely stated in Proposition 1 below. In this proposition, we do not assume a linear approximation architecture and a general correction function $\mathcal{K}_{\theta_t}$ is considered.

Suppose momentarily that the risk measure ρ admits the dynamic programming equations

$$Q_t(s,x) \stackrel{\text{def}}{=} C_t(s,x) + \rho(V_{t+1}(S_{t+1}^M(s_t, x, W_{t+1})) \mid s_t = s), \qquad (7)$$

$$V_t(s) = \min_{x \in \mathcal{X}_t(s)} Q_t(s,x), \quad t = T-1, \ldots, 0, \qquad (8)$$

where $V_T(S_T) \equiv 0$. This happens for example when ρ is the expectation, or the maximum over the possible realizations of the cost to go, see, e.g., Bertsekas (2005), Riedel (2004), Shapiro (2012). As a result, the notion of exact stochastic dynamic programing solution will be well-defined.

Proposition 1. *Suppose the risk functional ρ admits dynamic programming Equations (7) and (8). Let the feasible sets be defined by linear inequalities,*

$$\mathcal{X}_t(s_t) = \{x \in \mathbb{R}^n : A(s_t)x \leq b(s_t)\}, \qquad (9)$$

for some state-dependent matrix $A(s_t)$ and vector $b(s_t)$. Suppose that for every $s \in \mathcal{S}_t$, the functions $Q_t(s,x)$ are continuously differentiable and convex in x on $\mathcal{X}_t(s)$. In addition, assume that there exists a set of parameters $\{\bar{\theta}_0, \ldots, \bar{\theta}_{T-1}\}$ in $\Theta_0 \times \cdots \times \Theta_{T-1}$ such that

- *the functions $C_t(s,x) + \mathcal{K}_{\bar{\theta}_t}(s,x)$ are continuously differentiable in x, for each $s \in \mathcal{S}_t$, and*
- *the following equality on the gradient of the Q-function (7) and the gradient of the correction function holds at $\bar{x}_t(s) \stackrel{\text{def}}{=} X_t^\pi(s \mid \bar{\theta}_t)$:*

$$\nabla_x (C_t(s, \bar{x}_t(s)) + \mathcal{K}_{\bar{\theta}_t}(s, \bar{x}_t(s))) = \nabla_x Q_t(s, \bar{x}_t(s)), \quad \forall s \in \mathcal{S}_t. \qquad (10)$$

Then, $\{\bar{x}_t\}_{t=0}^{T-1}$ is an exact optimal stochastic dynamic programming solution.

A proof for Proposition 1 is provided in Appendix A.

Proposition 1 establishes a sufficient condition for the correction term to ensure that an optimal policy is included in the search problem. Admittedly, this condition might not be easy to verify since we do not have the optimal Q-function, but the existence of this sufficient condition implies that an appropriate correction term does not need to be equal to the value function.

The class of policies in (3) for the expectation is referred to as cost function approximation in Powell (2011). Other popular classes of policies in ADP are policies based on value function approximations and policy function approximations. In the former class, the value function in Bellman's equation is approximated using a linear model with parameters attempting to approximate the value of being in a state. In contrast to this class of policies, the correction term in the cost function approximation policies is not intended to approximate the value function and is being tuned only to produce a better policy. Policy function approximation-based policies are analytic functions, mapping states to actions, without using an embedded optimization problem. Parametric affine functions or polynomials belong to this class of policies. While parametric analytic functions can be used in the framework of direct policy search, when the feasible set $\mathcal{X}_t(s_t)$ includes a large number of inequality constraints on decisions $X_t^\pi(s_t)$, direct policy search for policy function approximation policies poses limitations, see, e.g., Moazeni et al. (2016).

In the rest of this section, we investigate the closeness to optimality of the policy obtained by the direct policy search, where the policy function approximation is expressed as in (3). Similar to the existing error bounds for the ADP solution (see, e.g., Bertsekas and Tsitsiklis 1996, Munos 2003, Yu and Bertsekas 2010), we consider a stationary MDP evolving over infinite-time horizon on a state-space $\mathcal{S}$, with the action space $\mathcal{X}$ and the cost function $C(s_t, x_t)$ to minimize an expected discounted cost objective. Under this assumption, the optimal policy corresponding to the fixed point of the Bellman operator is stationary; whence θ is time invariant. For an infinite-time horizon MDP, the direct policy search solves the following problem:

$$\min_{\theta\in\Theta} \mathbb{E}\left(\sum_{t=0}^{\infty} \gamma^t C(S_t, X^\pi(S_t \mid \theta)) \;\middle|\; S_0 = s \right). \tag{11}$$

Here, $\gamma < 1$ is a discount factor, and $X^\pi(S_t \mid \theta)$ is defined as in (3).

Suppose we are given a finite collection of basis functions $\{\phi_k\colon \mathcal{S} \to \mathbb{R}\}_{k=1}^K$ to specify the correction function behind the expectation over the next state. Let the hypothesis space of the policies in (3) be described by

$$\begin{aligned}\mathcal{H}^\pi = \big\{ &\pi_\theta\colon \mathcal{S} \to \mathcal{X} \text{ such that } \pi_\theta(s) \\ &\in \arg\min_{x\in\mathcal{X}(s)} \{C(s,x) + \gamma \mathbb{E}_{S'|s,x}(\theta^\top \Phi(S'))\}, \\ &\theta \in \Theta \big\}. \end{aligned} \tag{12}$$

Here, $\Phi(S')$ is the column vector with the basis functions ϕ_k evaluated at the next state S'. Denote the hypothesis space for the value functions, induced by the basis functions Φ and the parameter space Θ by $\mathcal{H}^V$:

$$\mathcal{H}^V \stackrel{\text{def}}{=} \{V\colon \mathcal{S} \to \mathbb{R} \text{ such that } V = \theta^\top \Phi,\ \theta \in \Theta\}. \tag{13}$$

Let Π_ξ denote a projection operator, that maps an arbitrary function $f\colon \mathcal{S} \to \mathbb{R}$ to the closest element of the hypothesis space $\mathcal{H}^V$, according to certain weighted norm $\|\cdot\|_\xi$. Therefore

$$\Pi_\xi f = \arg\min_{v\in\mathcal{H}^V} \|v - f\|_\xi = \theta_{\xi,f}^\top \Phi. \tag{14}$$

If ξ is the steady-state probability vector of the Markov chain under the policy being evaluated, then $\Pi_\xi T$ is a contraction mapping and has a fixed point denoted by $\bar{V}$ (see Proposition 6.3.1 of Bertsekas 2012), i.e., $\bar{V} = \Pi_\xi T \bar{V}$. Here, T is the Bellman operator, i.e.,

$$(TV)(s) \stackrel{\text{def}}{=} \min_{x\in\mathcal{X}(s)} C(s,x) + \gamma \mathbb{E}_{S'|x,s}[V(S')].$$

Projection approaches and linear approximation architectures are used, for instance, in approximate policy iteration, see, e.g., Bertsekas and Tsitsiklis (1996), Powell (2011), Bertsekas (2012).

The following proposition provides an error bound on the expected value of the policy obtained using direct policy search (11). Note that the value of the policy is a notion well distinct from the approximate value function that this policy could use to produce decisions at each state it visits. Let $\theta_{\xi,\bar{V}} \stackrel{\text{def}}{=} \arg\min_\theta \|T\bar{V} - \theta^\top \Phi\|_\xi$. In the following Proposition, $V^{\pi_{\theta_{\xi,\bar{V}}}}$ denotes the true value of the policy $\pi_{\theta_{\xi,\bar{V}}} \in \mathcal{H}^\pi$, which selects decisions according to $\pi_{\theta_{\xi,\bar{V}}}(s) = \arg\min_{x\in\mathcal{X}(s)} \{C(s,x) + \gamma \mathbb{E}_{S'|s,x}[\theta_{\xi,\bar{V}}^\top \Phi(S')]\}$.

Proposition 2. *Let V^* denote the optimal value function of the exact stochastic dynamic optimization, i.e., $V^* = TV^*$, and let $V^{\pi_{\theta^*}}(s)$ be the value of the policy in the policy hypothesis space $\mathcal{H}^\pi$ produced by the direct policy search method, where the policy is described as $\pi_{\theta^*}(s) = \arg\min_{x\in\mathcal{X}(s)} \{C(s,x) + \gamma \mathbb{E}_{S'|s,x}[\theta^{*\top} \Phi(S')]\}$, and θ^* is selected by optimizing (11). Then, the loss of optimality $V^{\pi_{\theta^*}}(s) - V^*(s)$ satisfies*

$$0 \le V^{\pi_{\theta^*}}(s) - V^*(s) \le V^{\pi_{\theta_{\xi,\bar{V}}}}(s) - V^*(s), \tag{15}$$

where $V^{\pi_{\theta_{\xi,\bar{V}}}}$ is the true value of the policy $\pi_{\theta_{\xi,\bar{V}}}$. Additionally, if the fixed point of the projected Bellman operator satisfies $V^{\pi_{\theta_{\xi,\bar{V}}}}(s) = \bar{V}(s) = \Pi_\xi T\bar{V}(s)$, then

$$\begin{aligned} V^{\pi_{\theta^*}}(s) - V^*(s) &\le \|V^* - \Pi_\xi T\bar{V}\|_\infty \\ &\le \frac{1}{1-\gamma} \|V^* - \Pi_\xi V^*\|_\infty. \end{aligned} \tag{16}$$

A proof for Proposition 2 is provided in Appendix B.

This bound is determined by the discount factor γ and supremum norm of the approximation error of the projected Bellman equation. Inequality (15) remains valid when the correction function includes the expectation; for further discussion, see the remark in Appendix B.

3.2. Interpolating Surface Model Approximation: Level of Nonstationarity

To manage the size of problem (6), for every $k = 1, \ldots, K$ in Equation (3), we let the tuning parameters $(\theta_{0,k}, \theta_{1,k}, \ldots, \theta_{T-2,k})$ follow an interpolating response surface model with $(\tau + 1)$ interpolation knots $\mathcal{T}_k \stackrel{\text{def}}{=} \{t_{0,k} = 0,\ t_{1,k}, \ldots, t_{\tau-1,k}, t_{\tau,k} = T-2\}$, placed with respect to time on the interval $[0, T-2]$, i.e.,

$$\theta_{t,k} = \mathcal{J}_{\mathcal{T}_k}(t), \quad t = 0, \ldots, T-2. \tag{17}$$

We refer to τ as the *level of nonstationarity*. The choice of $\tau = T-2$ is equivalent to a full nonstationary policy, whereas $\tau = 0$ implies a stationary policy. This feature allows the model user to adjust the stationarity of the policy depending on the available resources to solve the resulting stochastic optimization problem. In our numerical investigation, we consider splines with equidistant interpolation nodes, i.e., we let each $\theta_{t,k}$ be the value of a (piecewise-polynomial real) spline function $\mathcal{J}_{\mathcal{T}_k}(t)$, i.e.,

$$\mathcal{J}_{\mathcal{T}_k}(t) \stackrel{\text{def}}{=} \begin{cases} \mathcal{J}_{1,k}(t) & t_{0,k} \le t \le t_{1,k} \\ \vdots & \\ \mathcal{J}_{\tau,k}(t) & t_{\tau-1,k} \le t \le t_{\tau,k}. \end{cases} \tag{18}$$

Each $\mathcal{I}_{l,k}(t)$ is a polynomial function from $[t_{l-1,k}, t_{l,k})$ to $\mathbb{R}$. The highest order of these polynomials is said to be the order of the spline $\mathcal{I}_{\mathcal{T}_k}(t)$. The most commonly used splines are cubic splines, i.e., each $\mathcal{I}_{lk}(t)$ is a cubic polynomial

$$\mathcal{I}_{l,k}(t) = a_{l,k}(t - t_{l,k})^3 + b_{l,k}(t - t_{l,k})^2 + c_{l,k}(t - t_{l,k}) + d_{l,k}.$$

In our computational investigation in Section 5, we focus on natural cubic splines, with equidistant sets of interpolation knots, i.e., $t_{l,k} = ((T-2)/\tau)l$.

Each spline function $\mathcal{I}_{\mathcal{T}_k}$ is uniquely determined by its values at the spline knots. Hence, to determine $\mathcal{I}_{\mathcal{T}_k}(t)$, and consequently, $\theta_{t,k}$, it suffices to find the values $y_{0,k}, \ldots, y_{\tau,k}$ of the splines at their $\tau+1$ knots, i.e., $\mathcal{I}_{\mathcal{T}_k}(t_{l,k}) = y_{l,k}$ for $l = 0, \ldots, \tau$ and $k = 1, \ldots, K$. Therefore, using the approximation (17), the number of decision variables in the optimization problem (6) is reduced to $K \times (\tau + 1)$.

Response surface models have found great use as approximating functions in various contexts. The reader is referred to Jones (2001) for an excellent overview and comparison of surface models. Interpolating response surface models are more favorable than noninterpolating surfaces, because they are often more reliable and can better capture the shape of the function. Furthermore, in contrast to noninterpolating surfaces, for interpolating methods such as the natural cubic splines adding additional points often leads to a more accurate surface, which eventually converges to the true function. In stochastic dynamic programming, response surface models have often been used to approximate the future value functions between some discrete points in the state space for stochastic dynamic programming problems with continuous state and decision variables, e.g., Johnson et al. (1993) employs tensor product splines and Chen et al. (1999) adopts a method based on multivariate adaptive regression splines for approximating the future value functions, in the contexts of water resources application and inventory forecasting problem, respectively. In Coleman et al. (1999), a two-dimensional spline functional is used to directly approximate a local volatility function.

In contrast to the use of surface models for estimating the value function, in this paper, an interpolating surface model is adopted to approximate the pattern of decision variables and project the feasible space of problem (6) into the one with a lower dimension. While this method has been used in some works such as Coleman et al. (2007) to approximate the optimal policy $\mu_t(S_t)$ itself, to the best of our knowledge, its application to estimate nonstationary policy parameters is novel.

Our developed approach involves three approximation aspects; namely, approximating the policy function, approximating the correction functions through architecture of basis functions, and approximating the tuning parameters with an interpolating response surface model. Solving the resulting search problem (6) is discussed next.

3.3. Nested Hybrid Parallel Derivative-Free Optimization

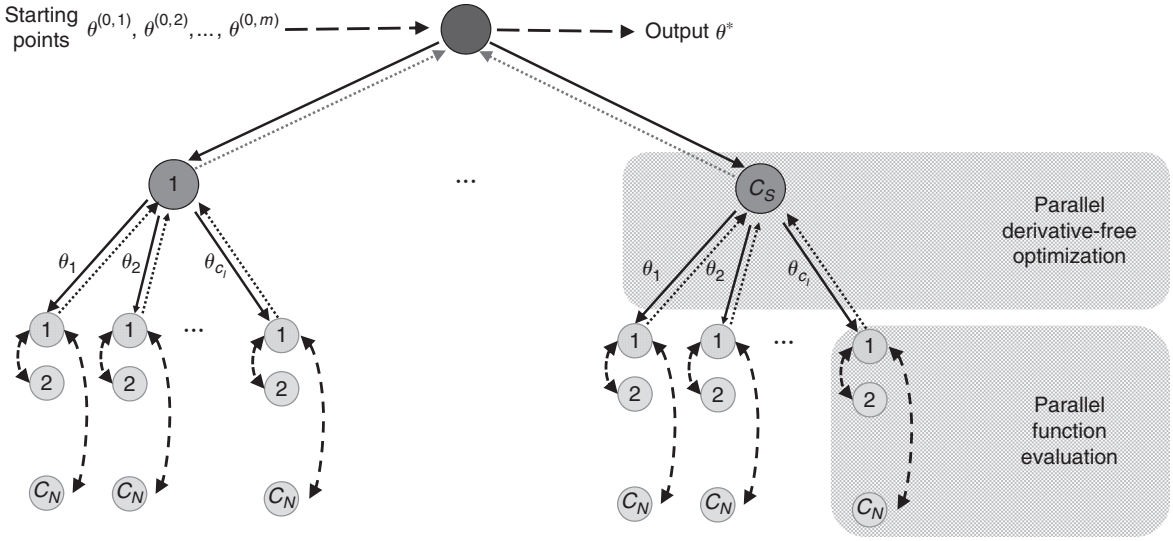
One common practice to estimate the expected value or risk in the objective function $\rho(B^\pi(S_0))$ in problem (6) is to draw N scenarios for exogenous state variables and approximate the criterion around these scenarios. This is also akin to generating N Monte Carlo trajectories or calling a generative model as in Ng and Jordan (2000). When N sample trajectories are used to approximate the objective function, each function evaluation $\rho(B^\pi(S_0))$ corresponding to a set of policy parameters requires solving $N \times T$ minimizations of type (3). This is computationally rather expensive.

To overcome this computational intensity, we take advantage of high performance computing. Our parallel model includes three levels of parallelism with $1 + c_S(1 + c_I c_N)$ processes. The processors' assignment for the developed parallel programming model is depicted in Figure 1.

In the first level, the *master processor* receives a set of starting points $\{\theta^{(0,1)}, \ldots, \theta^{(0,m)}\}$ and distributes them in subsets among c_S *manager nodes*. The master processor eventually collects the m computed solutions from the manager nodes and returns the best solution θ^*. In the second level, each manager processor receives the corresponding initial points, implements a parallel derivative-free optimization for each starting point one at a time, records the computed optimal points, and sends them back to the master node. Each manager node is provided with $c_I c_N$ *worker nodes* to do the optimization process. The manager node is also responsible for reconstructing the solution from the interpolating response surface model. The function evaluations at search directions are done in parallel by c_N worker processors, each of which solves a subset of the $N \times T$ minimizations of type (3). This constitutes the third layer of parallelization. The manager node communicates only with one of these c_N worker nodes, to which we refer as the *head worker node*. Each head worker processor assigns trajectories to worker nodes in its group, collects all of the solutions from them, evaluates the objective function of problem (6), and returns its value to its corresponding manager processor. The communications among processes in the first and second layers of parallelization are message passing, while the third layer of parallelism is expressed by multithreading.

One of the derivative-free optimization methods that can be perfectly parallelized is the multistart pattern search framework combined with Monte Carlo simulation. Pattern search methods choose a certain set of

Figure 1. Parallel Programming Model for Nonstationary Direct Policy Search



search directions at each iterate and search for a point that improves the objective function, see, e.g., Nocedal and Wright (2006). If a point with a significantly lower function value is found, it is adopted as the new iterate, and the center of the frame is shifted to this new point. This approach can be simply and effectively implemented on a parallel processing system, by evaluating the objective function at different directions simultaneously, see, e.g., Hough et al. (2001), Kolda (2005). The parallel pattern search we use in this paper is demonstrated in Figure 2. Here, γ^{tol} indicates the convergence tolerance, $0 \le \beta_c \le 1$ is the contraction factor, $\beta_e \ge 1$ is the expansion factor, the function $\varsigma(\cdot)$ is the sufficient decrease function from $[0,\infty)$ to the set of real numbers, $\mathcal{D} = \{d_i\}_i$ is the set of search directions, γ_0 is the $|\mathcal{D}|$-vector of initial step sizes, and $I^{\max}$ denotes the maximum number of iterations. In Figure 2, $\gamma_{l,i}$ refers to the ith element of the vectors γ_l. We note that $|\mathcal{D}|$ is not necessarily equal to c_I, and the candidate points $\{\theta^{(l+1,i)}\}_{i=1}^{|\mathcal{D}|}$ will be distributed almost equally among the head worker nodes. In addition, the points $\theta^{(\ell)}$ and $y^{(\ell)}$ are understood to be vectorized versions of $\{\theta^{(\ell)}_{t,k}\}_{t=0,\ldots,T-2,k=1,\ldots,K}$ and $\{y^{(\ell)}_{t,k}\}_{t=0,\ldots,\tau,k=1,\ldots,K}$.

Global convergence and robustness characteristics of some variations of this class of methods have been discussed in Torczon (1997), Lewis et al. (1998, 2000), Audet and Dennis (2002). In addition, the success of multistart direct search algorithms for a variety of test problems has been reported in Rios and Sahinidis (2013). The global convergence to a stationary point of the generalized pattern search method for an unconstrained problem with a continuously differentiable objective function is established in Torczon (1997). Audet and Dennis (2002) prove the existence of a limit point for any generalized pattern search method iteration, even if the objective function is not continuous. Further global convergence properties are analyzed in Dolan et al. (2003).

This parallel implementation to solve problem (6) is employed and assessed in connection with an optimal energy storage management problem, described in Section 4.

4. Application: Optimal Storage Charging Management

We adopt the model studied in Moazeni et al. (2015), in which a renewable power generation source (e.g., wind) and an energy storage device are integrated into an electrical grid to minimize the total cost of serving a demand over a finite-planning horizon $[0,T)$.

We discretize the time horizon $[0,T)$ into T intervals of length $\Delta t = 1$. The state of the system at time t is expressed by the state variable $S_t \stackrel{\text{def}}{=} (\tilde{D}_t, \tilde{E}_t, \tilde{P}_t, R_t)$, where $\tilde{D}_t$, $\tilde{E}_t$, and $\tilde{P}_t$, respectively, refer to the exogenously varying energy demand, energy supply, and energy price. Here, R_t is the fraction of the storage that is full at time t. The initial state $S_0 = (\tilde{D}_0, \tilde{E}_0, \tilde{P}_0, R_0)$ is known.

The energy supply $\tilde{E}_t$ is first used to serve the demand $\tilde{D}_t$. Given the state variable S_t, the decision to be made at the beginning of the time interval $[t, t+1)$ includes energy flows, $x_t^{GD}, x_t^{WG}, x_t^{WR}, x_t^{GR}, x_t^{RG}$, and x_t^{RD}, where x_t^{IJ} denotes the amount of energy transferred from I to J at time step t. The superscript W stands for wind, D for demand, R for storage, and G for grid. These nonnegative energy flows should satisfy the following constraints:

$$x_t^{WR} + x_t^{WG} = \tilde{E}_t - \min\{\tilde{E}_t, \tilde{D}_t\}, \tag{19}$$

$$x_t^{GD} + x_t^{RD} = \tilde{D}_t - \min\{\tilde{E}_t, \tilde{D}_t\}, \tag{20}$$

Figure 2. Parallel Nonstationary Direct Policy Search with the Level of Nonstationarity τ, and Parallel Pattern Search as the Parallel Derivative-Free Optimization Subroutine

Require: $\gamma^{\text{tol}}, \beta_c, \beta_e, \varsigma(\cdot), \mathscr{D} = \{d_i\}_i, \gamma_0, I^{\max}, \{\theta^{(0,1)}, \ldots, \theta^{(0,m)}\}, \Theta_t = [\theta^{\min}, \theta^{\max}]$ for $0 \le t \le T-2$.

(00) master processor sends the initial points $\theta^{(0,(n-1)\lfloor m/c_S \rfloor+1)}, \ldots, \theta^{(0,n\lfloor m/c_S \rfloor)}$ to manager nodes $n = 1, \ldots, c_S - 1$, and sends $\theta^{(0,(c_S-1)\lfloor m/c_S \rfloor+1)}, \ldots, \theta^{(0,m)}$ to the manager processor c_S.

(01) **for** $n = 1, 2, \ldots, c_S$, **do**

(02) **manager processor** n **receives the starting points and does the following:**

(03) **for** $s = (n-1)\left\lfloor \frac{m}{c_S} \right\rfloor + 1, \ldots, n\left\lfloor \frac{m}{c_S} \right\rfloor$, **do**

[invoking some parallel derivative-free optimization subroutine]

(04) initialize $\ell = 0$, $\theta^{(\ell)} = \theta^{(0,s)}$, and compute $y^{(\ell)}$ from $\theta^{(\ell)}$ at spline knots.

(05) $f^{(\ell)} = \rho\left(\sum_{t=0}^{T-1} C_t(S_t, X_t^{\pi}(S_t \mid \theta^{(\ell)})) \mid S_0 = s\right)$, where $X_t^{\pi}(S_t \mid \theta)$ is as in Equation (3) with the correction function as in Equation (5).

(06) **while** $\|\gamma_\ell\|_2^2 > \gamma^{tol}$ **or** $\ell \le I^{\max}$, **do**

(07) **for** $i = 1, \ldots, |\mathscr{D}|$, **do**

(08) $y^{(\ell+1,i)} = y^{(\ell)} + \gamma_{\ell,i} d_i$

(09) manager node n uses the interpolation models $\{\mathscr{I}_{\mathscr{T}_k}\}_{k=1}^K$ to reconstruct $\theta^{(\ell+1,i)}$ from $y^{(\ell+1,i)}$.

(10) $\theta^{(\ell+1,i)} = \min(\max(\theta^{(\ell+1,i)}, \theta^{\min}), \theta^{\max})$

(11) **end for**

(12) manager processor n distributes the $|\mathscr{D}|$ candidate iterates $\theta^{(\ell+1,1)}, \ldots, \theta^{(\ell+1,|\mathscr{D}|)}$ among the c_I head worker processors. Head workers employ other $c_N - 1$ worker processes to evaluate $f^{(\ell+1,i)} = \rho\left(\sum_{t=0}^{T-1} C_t(S_t, X_t^{\pi}(S_t \mid \theta^{(\ell+1,i)})) \mid S_0 = s\right)$ and return to manager node n.

(13) **if** $f^{(\ell+1,i)} < f^{(\ell)} - \varsigma(\gamma_{\ell,i})$ for some $d_i \in \mathscr{D}$, **do**

(14) $y^{(\ell+1)} = y^{(\ell)} + \gamma_{\ell,i} d_i$, $f^{(\ell+1)} = f^{(\ell+1,i)}$, and $\gamma_{\ell+1,i} = \beta_e \gamma_{\ell,i}$

(15) **else**

(16) $y^{(\ell+1)} = y^{(\ell)}$ and $\gamma_{\ell+1,i} = \beta_c \gamma_{\ell,i}$ for all $i = 1, \ldots, |\mathscr{D}|$.

(17) **end if**

(18) $\ell = \ell + 1$

(19) **end while**

(20) manager processor sets and records $\theta^{(*,s)} = \theta^{(\ell+1)}$, $f^{(*,s)} = f^{(\ell+1)}$.

(21) **end for**

(22) manager processor n sends solutions $\theta^{(*,(n-1)\lfloor m/c_S \rfloor+1)}, \theta^{(*,(n-1)\lfloor m/c_S \rfloor+2)}, \ldots, \theta^{(*,n\lfloor m/c_S \rfloor)}$, and their objective values $f^{(*,s)}$ to the master node.

(23) **end for**

(24) master processor receives $\theta^{(*,1)}, \theta^{(*,2)}, \ldots, \theta^{(*,m)}$ from manager nodes.

(25) master processor returns the point θ^* with minimum objective value among all.

(26) master processor broadcasts the termination of the algorithm to all other processors.

$$\frac{1}{\eta^D}(x_t^{RD} + x_t^{RG}) \le \Delta R^D R^{\mathbf{cap}}, \tag{21}$$

$$\eta^C(x_t^{GR} + x_t^{WR}) \le \Delta R^C R^{\mathbf{cap}}, \tag{22}$$

where $R^{\mathbf{cap}}$ denotes the maximum capacity of the storage device, and the *maximum charging rate* $\Delta R^C \in [0,1]$ and *maximum discharging rate* $\Delta R^D \in [0,1]$ are the maximum fraction of the storage, which can be charged or discharged, respectively, over the tth time interval. Here, $\bar{R}_t \overset{\text{def}}{=} R^{\max} - (1-\gamma_{\Delta t})R_t$ and $\underline{R}_t \overset{\text{def}}{=} (1-\gamma_{\Delta t})R_t - R^{\min}$, where $\gamma_{\Delta t} \in [0,1]$ is the dissipation loss constant. The nonnegative constants $R^{\min}$ and $R^{\max}$ indicate the minimum and maximum acceptable charge levels, which are percentages of the maximum capacity $R^{\mathbf{cap}}$. The constants $\eta^D \in (0,1]$ and $\eta^C \in (0,1]$ capture the charging efficiency and discharging efficiency rates, respectively. In a completely efficient storage device, we have $\eta^C = 1$ and $\eta^D = 1$.

Constraint (20) ensures that the demand is fully satisfied using the grid, the storage, or the wind turbine. Constraints (21) and (22) ensure that the storage device is not charged or discharged faster than allowed.

In addition to constraints (19)–(22), a feasible set of energy flows should imply that the storage level in the next time step, R_{t+1}, computed by

$$R_{t+1} \overset{\text{def}}{=} (1-\gamma_{\Delta t})R_t + \frac{1}{R^{\mathbf{cap}}}\left(\eta^C(x_t^{GR} + x_t^{WR}) - \frac{1}{\eta^D}(x_t^{RD} + x_t^{RG})\right), \quad t = 0, 1, \ldots, T-1, \tag{23}$$

satisfies $R_{t+1} \ge R^{\min}$ and $R_{t+1} \le R^{\max}$.

By using the equality constraints (19) and (20) to solve for the two decision variables x_t^{WR}, x_t^{RD} and incorporating appropriate constraints to ensure $x_t^{WR} \ge 0$, and $x_t^{RD} \ge 0$, the set of admissible decisions on energy flows over the time interval $[t, t+1)$ can be expressed in terms of $x_t = (x_t^{GR}, x_t^{GD}, x_t^{RG}, x_t^{WG})^\top$, the coefficient

matrix A, and the right-hand side vector $b(s_t)$, as follows:

$$\mathcal{X}_t(s_t) \stackrel{\text{def}}{=} \{x_t \in \mathbb{R}^4_+ : Ax_t \le b(s_t)\}.$$

Clearly, $\mathcal{X}_t(s_t)$ is a bounded and convex set, for every realization s_t of state S_t. The stage cost at time step t incurred by the system owner (decision maker) to fully serve the demand equals

$$C_t(S_t, x_t) \stackrel{\text{def}}{=} \tilde{P}_t x_t^{GR} + \tilde{P}_t x_t^{GD} - \tilde{P}_t \tilde{x}_t^{RG} - \tilde{P}_t x_t^{WG} - \tilde{P}_t \tilde{D}_t, \quad (24)$$

where $\tilde{P}_t \ge 0$ is the energy price. The decision maker's objective is addressed by $\rho(B^\pi(S_0))$, where $B^\pi(S_0)$ is as in Equation (2).

We consider stochastic processes as in Moazeni et al. (2015) to model the three sources of uncertainty in the problem. These models are described next.

4.1. Data Models

Energy Price $(\tilde{P}_t)$: The energy price (in \$/MWh) at time t is modeled by

$$\tilde{P}_t = P_t^{\text{hour}} + P_t^{\text{day}} + P_t^{\text{month}} + \exp(\tilde{Y}_t^P),$$

where the deterministic constants P_t^{hour}, P_t^{day}, and P_t^{month} address the hour of day, day of week, and month of year seasonality effects. The deseasonalized log-price $\tilde{Y}_t^P$ is estimated using a discretized mean-reverting jump diffusion process, with normally distributed jumps:

$$\tilde{Y}_t^P = \mu_P + (\tilde{Y}_{t-1}^P - \mu_P)e^{-\beta_P \Delta t} + \sigma_P \left(\frac{1 - e^{-2\beta_P \Delta t}}{2\beta_p}\right)^{1/2} \epsilon_t + \sum_{i=1}^{\tilde{q}_t} \tilde{J}_{t,i}.$$

Here, $\epsilon_t \sim \mathcal{N}(0,1)$, $\tilde{J}_{t,i} \sim \mathcal{N}(\mu_J, \sigma_J)$, and $\tilde{q}_t$ is a Poisson random variable with parameter λ_J.

Energy Demand $(\tilde{D}_t)$: The energy demand (in MWh) for the time interval $[t, t+1)$ is approximated by

$$\tilde{D}_t = D_t^{\text{hour}} + D_t^{\text{day}} + D_t^{\text{month}} + \tilde{Y}_t^D,$$

where D_t^{hour}, D_t^{day}, and D_t^{month} denote the hour of day, day of week, and month of year seasonal components, and the deseasonalized demands are modeled by the linear autoregressive (AR) model $\tilde{Y}_t^D = \phi_D \tilde{Y}_{t-1}^D + \sigma_D \tilde{\epsilon}_t$, where $\tilde{\epsilon}_t \sim \mathcal{N}(0,1)$. The constant parameters $|\phi_D| < 1$ and σ_D, and the initial demand $\tilde{D}_0$ are given.

The values of the parameters in the models for the price $\tilde{P}_t$ and demand $\tilde{D}_t$ have been estimated using the New York Independent System Operator data from 12 A.M. of January 1, 2007 (Monday) to 11:55 A.M. of December 31, 2011 (Saturday). These values are presented in Moazeni et al. (2015, Tables II–V).

Wind Energy $(\tilde{E}_t)$: We assume that the wind farm has 50 identical wind turbines, each of which has an output $p_{\tilde{W}_t}^{(1)}$ given by

$$p_{\tilde{W}_t}^{(1)} = \begin{cases} 0 & \text{if } \tilde{W}_t < 0 \text{ or } \tilde{W}_t > 25 \\ 10^{-6} \times \frac{1}{2} A \nu C_p \tilde{W}_t^3 & \text{if } 0 \le \tilde{W}_t < v_r \\ p_r & \text{if } v_r \le \tilde{W}_t \le 25, \end{cases} \quad (25)$$

where $\tilde{W}_t$ denotes the wind speed measured in m/s. Here, we let the rated power be $p_r = 4$ MW, the power coefficient $C_p = 0.5$, $A = 50^2\pi$ (in m^2), the density of air $\nu \approx 1.3$ kg/m^3, and the minimal wind speed corresponding to the rated power $v_r \approx 11.62$ m/s, see, e.g., MacKay (2009) and Moazeni et al. (2015). Velocity of the wind is assumed to be equal to $\tilde{W}_t = (\tilde{Y}_t^E + \mu_E)^2$, with given $\tilde{W}_0$ and μ_E. Here, $\tilde{Y}_t^E$ evolves according to an AR(1) model $\tilde{Y}_t^E = \phi_E \tilde{Y}_{t-1}^E + \sigma_E \sqrt{\Delta t}\, \tilde{\epsilon}_t$, where $\tilde{\epsilon}_t \sim \mathcal{N}(0,1)$ and the deterministic initial value $\tilde{Y}_0^E = 0$ and the given $|\phi_E| < 1$ and σ_E. Therefore the (total) wind energy output (measured in MWh) from the 50 wind turbines over the time interval $[t, t+1)$ (one hour) can be approximated by $50 \times p_{\tilde{W}_t}^{(1)}$.

4.2. Exact Stochastic Dynamic Programming Solution

In this section, we provide a closed-form representation for an exact stochastic dynamic programming solution for a stylized fully efficient storage device model, when solely the expected cost is concerned, i.e., the risk functional ρ is the expectation $\mathbb{E}$. This result is presented in Proposition 3.

We assume that all of the expectations and conditional expectations are well defined. Throughout, denote the vertical vector $(1, 1, -1, -1)^\top$ by e and let $\mathbb{E}_t[\cdot] \stackrel{\text{def}}{=} \mathbb{E}[\cdot \mid S_t]$. In addition, define

$$x_t^+ \stackrel{\text{def}}{=} (\bar{R}_t R^{\mathbf{cap}}, \tilde{E}_t - \min\{\tilde{E}_t, \tilde{D}_t\}, \tilde{D}_t - \min\{\tilde{E}_t, \tilde{D}_t\}, 0), \quad (26)$$

$$x_t^- \stackrel{\text{def}}{=} (0, \tilde{D}_t - \min\{\tilde{E}_t, \tilde{D}_t\}, \underline{R}_t R^{\mathbf{cap}}, \tilde{E}_t - \min\{\tilde{E}_t, \tilde{D}_t\}). \quad (27)$$

Proposition 3. *Let $\eta^D = \eta^C = 1$, $\Delta R^D = \Delta R^C = 1$, and $\gamma_{\Delta t} = 0$. Then at timesteps $t = 0, \ldots, T-2$, the function $Q_t(s, x)$ equals*

$$Q_t(s,x) = (\tilde{P}_t - \mathbb{E}[\tilde{P}_{t+1} \mid S_t = s])e^\top x - \mathbb{E}[\tilde{P}_{t+1} \mid S_t = s] \cdot (\underline{R}_t R^{\mathbf{cap}} + \tilde{E}_t - \tilde{D}_t) + B_t(R^{\max} - R^{\min})R^{\mathbf{cap}} - \sum_{i=t+1}^{T-1} \mathbb{E}[\tilde{P}_i \tilde{E}_i \mid S_t = s]. \quad (28)$$

where $B_{T-2} = 0$ and

$$B_t = \Pr[\tilde{P}_{t+1} \ge \mathbb{E}_{t+1}[\tilde{P}_{t+2}] \mid \tilde{P}_t] \cdot \mathbb{E}_t[(\tilde{P}_{t+1} - \mathbb{E}_{t+1}[\tilde{P}_{t+2}]) \mid \tilde{P}_{t+1} \ge \mathbb{E}_{t+1}[\tilde{P}_{t+2}]] + \mathbb{E}_t[B_{t+1}]. \quad (29)$$

In addition, for $t = 0, \ldots, T-2$, *an optimal decision is given by*

$$\begin{cases} x_t^+ & \text{if } \tilde{P}_t \le \mathbb{E}[\tilde{P}_{t+1} \mid S_t], \\ x_t^- & \text{if } \tilde{P}_t > \mathbb{E}[\tilde{P}_{t+1} \mid S_t]. \end{cases}$$

At the last time step $t = T-1$, *the myopic decision is optimal.*

A proof for Proposition 3 is provided in Appendix C.

Under the assumptions of Proposition 3, the decision x_t^- yields $R_{t+1} = R^{\min}$, while x_t^+ implies $R_{t+1} = R^{\max}$.

The derivation of the exact solution in Proposition 3 relies on strong assumptions on the characteristics of the storage device. In practice, often $\eta^D \eta^C < 1$, $\Delta R^D \Delta R^C < 1$, and $\gamma_{\Delta t} > 0$. For more realistic storage models, finding an exact solution to the stochastic dynamic programming problem is, in general, difficult. Solving such problems numerically can become quite challenging when the dimension of the state space increases and actions are constrained; hence we should resort to approximation techniques such as the direct policy search from ADP.

5. Computational Experiments

In this ection, we investigate the performance and limitations of the developed methodology and parallel implementations in terms of handling nonstationarity and *arbitrary* risk functionals $\rho(\cdot)$, for the optimal energy storage management application described in Section 4.

The computations are carried out partially on the *Hyperion* cluster from the Lawrence Livermore National Laboratory (https://hyperionproject.llnl.gov/index.php) and the *Tower* cluster of CASTLE Labs at Princeton University. The Java implementation uses the Java message-passing interface library *MPJ Express* (Shafi et al. 2009).

The base model in our experiments is formulated over a planning horizon of $T = 168$ hours, with $R^{\min} = 0.1$, $R^{\max} = 0.9$, $R_0 = 0.1$ and $R^{\mathbf{cap}} = 1{,}000$ [MWh] (capacity parameters), $\Delta R^C = 0.2$ and $\Delta R^D = 0.25$ (charge and discharge maximum rates), $\eta^C = 0.75$ and $\eta^D = 0.9$ (charge and discharge inefficiencies), and $\gamma_{\Delta t} = 0$ (no leakage). These values for the storage device are similar to those in Table VI in Moazeni et al. (2015). Full details on the parameter estimation methods and on the data sets for the models in Section 4.1 are provided in the online supplement to this paper.

A policy based on a cost function approximation $X_t^\pi(s_t \mid \theta_t)$ is considered, as in Equation (3). The stage cost $C_t(s_t, x_t)$ is given by (24). A linear approximation architecture of the form (5) is adopted for the correction function:

$$\mathcal{K}_{\theta_t}(s_t, x_t) = \theta_{t,1} \phi_1(s_t, x_t). \tag{30}$$

This approximation uses a single basis function ϕ_1, defined as

$$\begin{aligned} \phi_1(s_t, x_t) = (-\eta^D (1 - \gamma_{\Delta t}) R^{\mathbf{cap}} R_t - \eta^C \eta^D (x_t^{GR} + x_t^{WR}) \\ + (x_t^{RD} + x_t^{RG})) \mathbb{E}[\tilde{P}_{t+1} \mid S_t = s_t]. \end{aligned} \tag{31}$$

The time-varying weights $\theta_{t,1}$ can take values in $[-2, 4]$. The variability of $\theta_{t,1}$ over time is controlled by the level of nonstationarity parameter τ. At the last stage $\mathcal{K}_{\theta_{T-1}}(S_{T-1}, \theta_{T-1})$ is set to 0, indicating that the policy will behave myopically. For the analyses reported in this section, the conditional expectation in (31) is computed in closed form for the price model in Section 4.1.

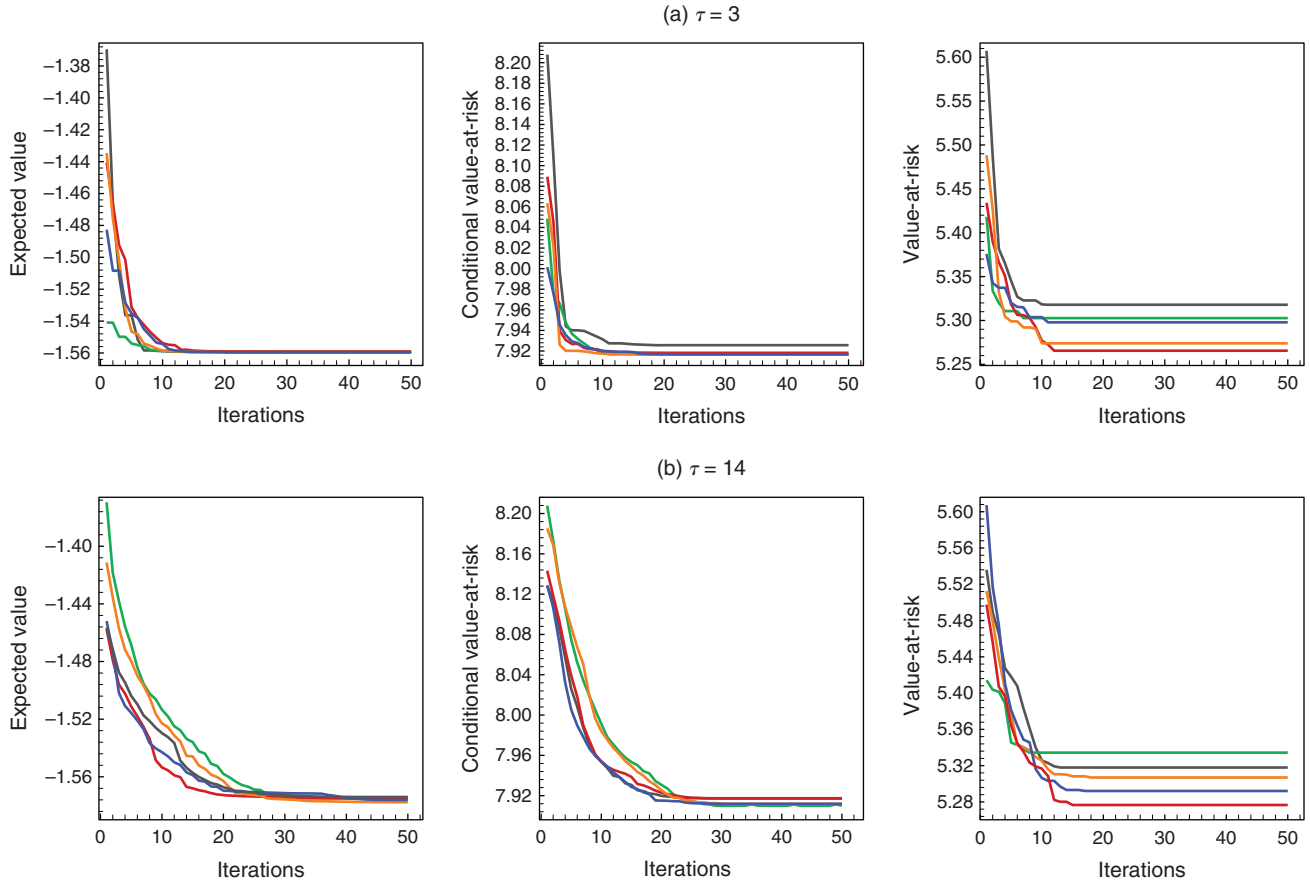
The basis function (31) is found to be able to produce a range of objectives that can give more or less importance to discharging now versus discharging later, depending on the value of the weight $\theta_{t,1}$. For example, with $\theta_{t,1} = 0$, the one-stage cost minimization is the only goal, resulting in a tendency to discharge quickly. With a large value for $\theta_{t,1}$, the objective essentially reduces to maximizing the next storage-level R_{t+1}, resulting in a tendency to charge as much as possible to be able to discharge at the next time step.

We study three risk functionals for the objective function in the direct policy search: $\mathbb{E}[\cdot]$ (the expectation), VaR_β (the VaR at confidence-level β), and CVaR_β the conditional VaR at confidence-level β with β set to 95%.

When the risk functional is the expectation, we know the optimal value of the problem, as established in Proposition 3. Therefore the suboptimality of the solutions produced with our methodology can be rigorously assessed (see Section 5.1). We are not making any claim that our proposed methodology is computationally more effective than other ADP methods optimized for that purpose. When the objective function is the VaR or the CVaR, there does not exist a stochastic dynamic programming reformulation able to minimize those objectives. As a result, it would not be possible to apply other standard ADP methods, which rely on the existence of the recursive dynamic programming equations, for the purpose of comparing relative performances.

For the implementation of the parallel multistart pattern search, we use $N = 10{,}000$ simulated paths. The initial step sizes are set to $\gamma_{0,i} = 1.5$ for all directions i. The sufficient decrease function $\varsigma(\cdot)$ that is set to the constant-valued function 0.1. The expansion parameter β_e and the contraction parameter β_c that control the relative change of the step sizes are set to $\beta_e = 2$ and $\beta_c = 0.5$, respectively. Those choices are typical in the literature, see, e.g., Nocedal and Wright (2006). The tolerance parameter specified by the stopping criterion is set to $\gamma^{\text{tol}} = 10^{-3}$ and the maximum number of iterations is set to $I^{\max} = 25$.

Figure 3. (Color online) Improvement in the Objective Function of Problem (6) Over Optimization Iterations (y-Axis, in Millions)



Notes. In each plot, each curve indicates the objective function improvements associated with a randomly generated initial point θ_0 for the optimization.

To evaluate the impact of the stopping criterion, we conducted an analysis of the improvement in the objective function over 50 iterations, for five different random starting points and for the special value 0. We replicated the experiment for two values of the level of nonstationarity τ. The simulation results are presented in Figure 3.

The conclusion of this study is that in all cases, 25 iterations are enough to obtain a solution reasonably close to solutions obtained with more iterations. In general, with a higher level of nonstationarity, the maximum number of iterations should increase. We also observe that the objective function stops improving sooner with the VaR criterion. This suggests that with the VaR, it is more effective to choose a tighter stopping criterion and try more initial starting points. For the expectation and CVaR, it is expected that the search algorithm should be less sensitive to the starting points, thanks to the convergence properties resulting from the convexity of those functionals.

Next, we investigate the impact of the number of scenarios N on the single-thread computational time for the objective evaluation. Table 1 reports the averaged time required to evaluate each objective function $\rho(B^\pi(S_0))$, over 200 nonstationary policies, for the three risk functionals and for different numbers of samples. We generated the policy parameters randomly, and replicated the experiment for two values of the parameter τ. Our conclusion is that the computational time for each objective function evaluation grows more or less linearly with the number of simulated paths, and this holds true for each of the three risk functionals we studied.

5.1. Quality of the Policy: Accuracy and the Impact of Level of Nonstationarity

We first address the question of evaluating the quality of the computed policies. We start by using the special case of the energy storage charging problem, for which a closed-form representation of the optimal solution is established in Proposition 3. Namely, we set $\eta^D = 1$, $\eta^C = 1$, $\gamma_{\Delta t} = 0$, and $\Delta R^D = 1$, while the other parameter values remain as before. According to Proposition 3, our choice of the correction functional $\mathcal{H}_t$ in Equations (31) and (10), the optimal parameter values

Table 1. Average Single-Thread Runtimes to Evaluate the Objective Function of Problem (6)

(a) $\tau = 3$				(b) $\tau = 14$			
	Objective				Objective		
N	$\mathbb{E}$	$\text{CVaR}_{95\%}$	$\text{VaR}_{95\%}$	N	$\mathbb{E}$	$\text{CVaR}_{95\%}$	$\text{VaR}_{95\%}$
100	11.45	11.87	12.16	100	11.44	11.48	11.49
500	78.69	78.22	80.03	500	58.04	57.84	57.91
1,000	115.85	116.09	114.56	1,000	115.37	113.84	114.58
5,000	585.84	583.22	579.11	5,000	581.39	572.96	574.96
10,000	1,144.85	1,133.29	1,161.69	10,000	1,153.12	1,148.52	1,167.97

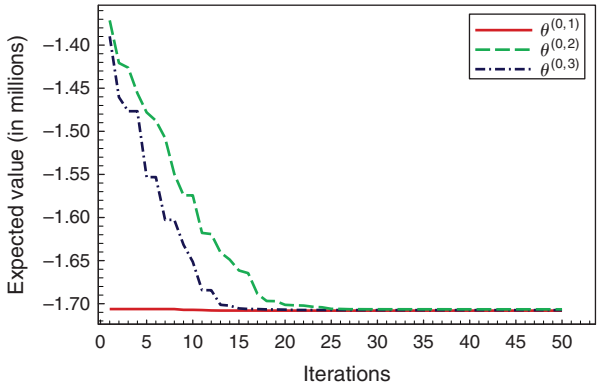
Note. The reported evaluation times are averaged over 200 randomly generated policy parameter vectors $(\theta_1, \ldots, \theta_{T-1})$.

must be $\theta_t^* = 1.0$ for $t = 0, \ldots, T-2$. We observe that our computational approach is able to recover the optimal parameter values $\theta_t^* = 1.0$ for $t = 0, \ldots, T-2$. In addition, to study whether the convergence to the optimal policy is robust with respect to the selected starting point, we conduct a sensitivity analysis relative to the initial starting points. Simulations from this experiment are reported in Figure 4. This study indicates that the developed approach consistently returns the optimal policy for the test case, where we could actually determine an optimal policy theoretically.

Next, we investigate the impact of the interpolating surface parameter τ on the computed policy and its corresponding objective value, when we are not able to establish and refer to an exact optimal policy. We focus on the relative improvement that can be achieved when optimizing over nonstationary policies for increasing levels of complexity of the policy space. Table 2 reports the results of our experiments for three values of the parameter τ that controls the potential nonstationarity of the policies and for the alternative risk functionals. The improvements are particularly visible on the VaR functional, which is in principle the most challenging risk measure among the three to optimize. Our conclusion is that increasing the level of nonstationarity, and thus the number of parameters to optimize, may significantly improve the quality of the policies; the achieved improvement depends on the risk functional. The runtime of the multithreading parallel implementation that tunes the stationary policy when $N = 10{,}000$ is around 88%–91% of the runtime of the nested hybrid parallel implementation that tunes the nonstationary policy, as reported in Table 5. This ratio is found to be consistent for the three objectives and for different values of τ.

The corresponding computed nonstationary policy parameters θ_t are illustrated in Figure 5. In these plots, the optimal parameter for the stationary policy is depicted by the straight thick red line. In Figure 5, left plots illustrate the nonstationary policy parameter curve, when the risk functional ρ is $\mathbb{E}$, and the right plots correspond to the nonstationary policy parameter curve, when the risk functional ρ is $\text{VaR}_{95\%}$. A comparison between the left plots and the right plots indicates that the deviation of the nonstationary policy parameter curve from the stationary policy parameter is larger for $\text{VaR}_{95\%}$ than for the expectation, at some timesteps. That can be the reason for the significant relative improvement in the objective function for $\text{VaR}_{95\%}$, when the policy is let be nonstationary.

Figure 4. (Color online) Iterations of the Algorithm of Figure 2 for the Expected Value Risk Functional (y-Axis, in Millions)



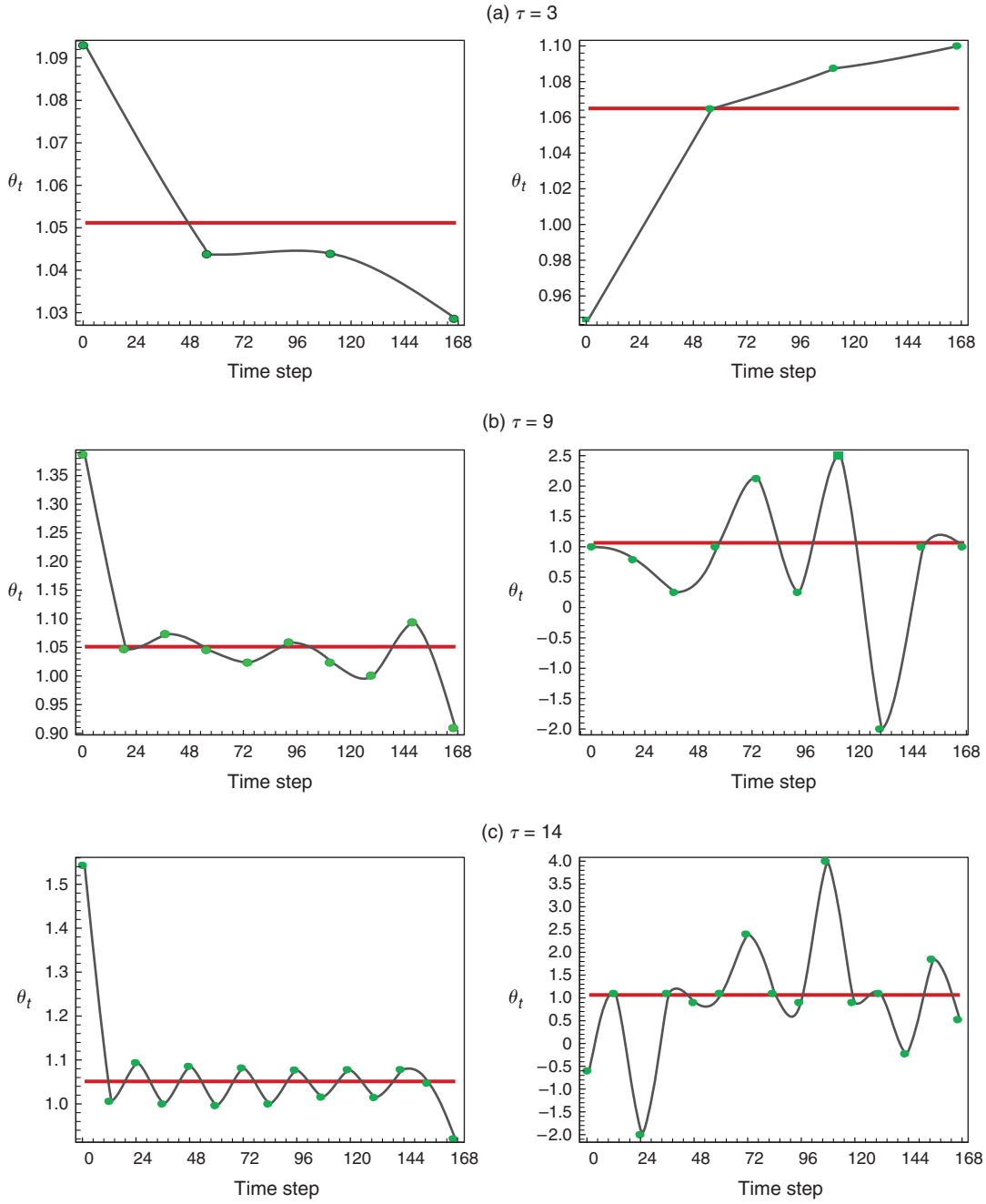
θ^*	$\theta^{(*,1)}$	$\theta^{(*,2)}$	$\theta^{(*,3)}$
1.0	1.01171	1.00858	1.03196
1.0	0.99991	1.00908	0.99909
1.0	1.00144	0.99312	1.00721
1.0	1.01079	1.03148	1.00878

Notes. Runs from the starting points $\theta^{(0,1)} = (1,1,1,1)$, $\theta^{(0,2)} = (0,0,0,0)$, and a randomly generated $\theta^{(0,3)} = (0.0417, 2.5799, 0.0734, 3.8421)$ converge to the exact solution θ^* established in Proposition 3.

Table 2. Relative Improvement in the Objective Function of the Nonstationary Policy for Several Values of τ

(a) Relative to the optimal stationary policy				(b) Relative to the myopic policy			
	Objective				Objective		
τ	$\mathbb{E}$ (%)	$\text{CVaR}_{95\%}$ (%)	$\text{VaR}_{95\%}$ (%)	τ	$\mathbb{E}$ (%)	$\text{CVaR}_{95\%}$ (%)	$\text{VaR}_{95\%}$ (%)
3	0.25	1.34	14.67	3	15.49	3.41	1,328.49
9	0.43	1.52	27.08	9	15.69	3.59	1,483.17
14	1.26	1.87	28.83	14	16.66	3.93	1,504.92

Figure 5. (Color online) Computed Policy Parameters $\theta_0, \ldots, \theta_{166}$ for Different Choices of τ



Notes. In left-hand plots the risk functional ρ is $\mathbb{E}$. In right-hand plots the risk functional ρ is $\text{VaR}_{95\%}$. In all plots, the straight thick red line indicates parameter of the optimal stationary policy. Green circles on the nonstationary policy parameter curve show the solutions at the spline knots.

5.2. Parallel Implementation Analyses

To evaluate our parallel implementation model for the three different nested levels of parallelism, as described in Section 3.3, we conduct several experiments explained.

To study the computational benefits brought by our multithreading implementation at the level of the objective evaluation, we first replicate the single-thread experiments of Table 1, this time with 32 threads. The results obtained on the Hyperion supercomputer are reported in Table 3. Compared to the single-thread implementation, we observe a reduction of objective evaluation times by roughly a fivefold factor.

Next, we explore the computational benefits of different types of parallel implementations; namely, the multithreading implementation at the level of objective evaluation parallelization ($c_S = 1$, $c_I = 1$, $c_N = 32$), the message-passing parallel implementation for the search optimization problem in which $c_I = 2(\tau + 1)$ processes communicate by message-passing while $c_N = 1$ and $c_S = 1$, and a hybrid model with nested layers of parallelization, including multithreading and message-passing ($c_I = 2(\tau + 1)$, $c_N = 32$, $c_S = 1$). Table 5 reports the improvements in the runtime of these parallel implementations relative to the sequential implementation computed by

$$100 \times \left(1 - \frac{\text{runtime of parallel implementation}}{\text{runtime of sequential implementation}}\right).$$

The runtimes of the sequential implementation with the single-thread model for the objective evaluations are reported in Table 4. Time limits for the sequential and multithread experiments are set to 86,340 seconds (= 23 hours 59 minutes) and 18,000 seconds (= 5 hours), respectively. This has resulted in the zeros in Table 5.

The results in Table 5 indicate that the multithreading implementation for the objective evaluations could reduce the computational time of the whole direct search approach by a factor around 5, as well. Improvement in computational time with message passing for the optimization step is more pronounced than the improvement realized from multithreading at the level of objective function evaluations. This observation suggests that when a limited number of processes are available, it would be more efficient to allocate them to the second level of parallelization than the third layer. In addition, while the runtime of the sequential implementation increases when the parameter τ increases, the runtime of the message-passing parallel implementation is almost in the same range of magnitude as τ and c_I increases. The results in the last block of Table 5 (nested hybrid) show that a parallel implementation with nested layers of parallelization can decrease the computational time to 5% (first row) to 30% (last row) of the sequential runtime.

Table 3. Study of the Evaluation Times in Terms of the Number of Samples, Using 32 Threads: Average Time Over 200 Randomly Generated Nonstationary Policies

(a) $\tau = 3$				(b) $\tau = 14$			
	Objective				Objective		
N	$\mathbb{E}$	$\text{CVaR}_{95\%}$	$\text{VaR}_{95\%}$	N	$\mathbb{E}$	$\text{CVaR}_{95\%}$	$\text{VaR}_{95\%}$
100	2.72	2.86	2.75	100	2.84	2.89	2.86
500	14.04	14.04	14.13	500	14.25	14.21	14.28
1,000	28.74	28.56	29.45	1,000	28.67	28.85	29.92
5,000	145.20	144.68	143.54	5,000	145.72	145.41	145.33
10,000	287.32	286.13	286.84	10,000	289.76	287.58	296.90

Table 4. Runtime (in Seconds) of Sequential Implementations of Direct Policy Search ($c_S = 1$, $c_I = 1$, $c_N = 1$)

Sequential implementation			
	τ		
N	3	9	14
$\mathbb{E}$			
100	2,624	6,575	8,956
500	17,856	30,816	45,074
1,000	26,229	60,618	>86,340
5,000	>86,340	>86,340	>86,340
$\text{CVaR}_{95\%}$			
100	2,908	6,113	8,988
500	17,730	30,567	44,923
1,000	26,298	61,480	>86,340
5,000	>86,340	>86,340	>86,340
$\text{VaR}_{95\%}$			
100	2,899	8,409	8,987
500	18,143	30,480	44,976
1,000	25,939	60,602	>86,340
5,000	>86,340	>86,340	>86,340

6. Concluding Remarks

The current literature on direct policy search methods for multistage dynamic optimization problems typically assumes that the policy parameters are stationary or that the search is restricted to the class of affine policies. In this work, we develop a computational approach to approximate nonstationary policy parameters for a fairly general class of policy approximators. A prominent feature of our cost function approximation policy is that it can handle a large number of constraints on the feasible actions, and thereby avoiding some limitations of affine policies in the context of constrained dynamic optimization problems. No assumption is made on the differentiability of the correction term or the objective function of the stochastic search problem, which makes the approach suitable for a large variety of risk measures, such as the VaR. The computational tractability of the framework follows from a nested hybrid parallel model including

Table 5. Relative Improvements [%] in Runtime of Different Parallel Implementations

	Parallel implementations								
	τ								
	Multithreading			Message passing			Nested hybrid		
N	3	9	14	3	9	14	3	9	14
$\mathbb{E}$									
100	71.00%	74.78	73.49	68.52%	87.54	93.21	85.40%	93.96	95.92
500	81.04	74.84	74.94	83.31	90.36	93.36	92.72	95.61	97.28
1,000	74.56	74.70	74.01	69.05	86.82	93.10	85.36	93.73	95.73
5,000	61.73	12.12	0	79.15	79.15	79.15	85.18	84.76	85.05
10,000	24.55	0	0	79.15	79.15	79.15	70.42	70.48	70.66
$\text{CVaR}_{95\%}$									
100	72.97	72.83	73.32	79.20	86.60	93.20	87.83	93.55	95.99
500	80.90	74.70	75.03	83.16	86.70	93.30	92.58	95.25	96.95
1,000	74.80	75.07	74.02	69.10	90.27	93.07	85.29	93.75	95.62
5,000	61.86	11.44	0	79.15	79.15	79.15	85.05	85.55	84.74
10,000	24.86	0	0	79.15	79.15	79.15	70.88	70.13	70.80
$\text{VaR}_{95\%}$									
100	73.85	79.83	73.52	79.13	92.81	90.80	88.13	95.43	95.74
500	81.39	74.55	73.52	77.92	86.77	93.37	92.87	95.38	97.29
1,000	74.43	73.84	73.25	68.85	90.14	93.10	85.04	93.70	95.62
5,000	62.18	11.41	0	79.15	79.15	79.15	84.92	84.75	84.81
10,000	24.68	0	0	79.15	79.15	79.15	71.15	69.85	70.91

Notes. In Multithreading Parallel Implementation $c_S = 1$, $c_I = 1$, and $c_N = 32$. In Message-Passing Parallel Implementation $c_S = 1$, $c_I = 2(\tau + 1)$, and $c_N = 1$. In Nested Hybrid Parallel Implementation $c_S = 1$ and $c_I = 2(\tau + 1)$, and $c_N = 32$.

the use of multistart parallel derivative-free optimization, and parallel function evaluations. The method and its efficiency are investigated on an energy storage charging management problem for three risk functionals. To model the trade-off between accuracy and computational effort, we represent the policy parameters through an interpolating response surface model over time and introduce the concept of level of nonstationarity. This allows the decision maker to adjust the nonstationarity-level based on the available resources.

Investigating optimal placement of interpolating knots remains as future work. In our implementation, we use an equal number of knots $\tau + 1$ for all k. However, one may choose different numbers of knots for different k; that is, $\tau_k + 1$. For example, if a given basis function seems more effective at approximating the policy function, we may choose a larger number of knots for that basis function, while selecting a smaller number of knots for other basis functions. Using other response surfaces to model nonstationary policy parameters suggests another future research direction. In this paper, we assume that the architecture for the correction term in the policy function approximation is given. It would be fruitful to investigate the impact of the specification of the architecture on the computed policy and its performance (Tsitsiklis and Roy 1996, Keller et al. 2006, Yu and Bertsekas 2009). Finally, further theoretical analyses on the cost function approximation might be possible and remain as future work. For instance, it would be of interest to be able to identify sufficient conditions on the correction term under which the policy approximator is reduced to an affine function of the policy parameters, beyond classical control problems where the value function is known to be quadratic in the state variables.

Acknowledgments

The authors thank the area editor and referees whose comments led to significant improvements in the paper.

References

Audet C, Dennis JEJ (2002) Analysis of generalized pattern searches. *SIAM J. Optim.* 13(3):889–903.

Baxter J, Bartlett PL (2001) Infinite-horizon policy-gradient estimation. *J. Artificial Intelligence Res.* 15(1):319–350.

Bertsekas DP (2005) *Dynamic Programming and Optimal Control*, Vol. 1 3rd ed. (Athena Scientific, Belmont, MA).

Bertsekas DP (2012) *Dynamic Programming and Optimal Control*, Vol. II, 4th ed., Approximate Dynamic Programming (Athena Scientific, Belmont, MA).

Bertsekas D, Tsitsiklis (1989) *Parallel and Distributed Computation: Numerical Methods* (Prentice-Hall, Englewood Cliffs, NJ).

Bertsekas DP, Tsitsiklis J (1996) *Neuro-Dynamic Programming*, 1st ed. (Athena Scientific, Belmont, MA).

Bertsimas D, Goyal V (2012) On the power and limitations of affine policies in two-stage adaptive optimization. *Math. Programming* 134(2):491–531.

Bertsimas D, Iancu DA, Parrilo PA (2010) Optimality of affine policies in multistage robust optimization. *Math. Oper. Res.* 35(2): 363–394.

Brown PD, Lopes JAP, Matos MA (2008) Optimization of pumped storage capacity in an isolated power system with large renewable penetration. *IEEE Trans. Power Systems* 23(2):523–531.

Burger M, Graeber B, Schindlmayr G (2008) *Managing Energy Risk: An Integrated View on Power and Other Energy Markets* (John Wiley and Sons, Chichester, UK).

Busoniu L, Babuska R, Schutter BD, Ernst D (2010) *Reinforcement Learning and Dynamic Programming Using Function Approximators*, 1st ed. (CRC Press, Boca Raton, FL).

Carmona R, Ludkovski M (2010) Valuation of energy storage: An optimal switching approach. *Quant. Finance* 10(4):359–374.

Censor Y, Zenios SA (1998) *Parallel Optimization: Theory, Algorithms, and Applications* (Oxford University Press, New York).

Chen VCP, Ruppert D, Shoemaker CA (1999) Applying experimental design and regression splines to high-dimensional continuous-state stochastic dynamic programming. *Oper. Res.* 47(1):38–53.

Coleman TF, Li Y, Patron C (2007) Total risk minimization using Monte Carlo simulations. Birge JR, Linetsky V, eds. *Handbooks in Operations Research and Management Science* (Elsevier, Amsterdam), 593–635.

Coleman TF, Li Y, Verma A (1999) Reconstructing the unknown local volatility function. *J. Comput. Finance* 2(3):77–102.

Defourny B, Ernst D, Wehenkel L (2008) Risk-aware decision making and dynamic programming. *Proc. 22nd Annual Conf. Neural Inform. Processing Systems, Vancouver, Canada*, 1–8.

Dixon LCW, Jha M (1993) Parallel algorithms for global optimization. *J. Optim. Theory Appl.* 79(2):385–395.

Dolan ED, Lewis RM, Torczon V (2003) On the local convergence of pattern search. *SIAM J. Optim.* 14(2):567–583.

Duffie D, Pan J (1997) An overview of value at risk. *J. Derivatives* 4(3):7–49.

Giuliani M, Castelletti A, Pianosi F, Mason E, Reed P (2015) Curses, tradeoffs, and scalable management: Advancing evolutionary multiobjective direct policy search to improve water reservoir operations. *J. Water Resources Planning Management* 142(2): Article no. 04015050.

Golub G, Ortega JM (1993) *Scientific Computing: An Introduction with Parallel Computing* (Academic Press, San Diego).

Gonzalez JG, de la Muela RMR, Santos LM, Gonzalez AM (2008) Stochastic joint optimization of wind generation and pumped-storage units in an electricity market. *IEEE Trans. Power Systems* 23(2):460–468.

Grondman I, Busoniu L, Lopes GAD, Babuska R (2012) A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Trans. Systems, Man, Cybernetics—Part C: Appl. Rev.* 42(6):1291–1307.

Harsha P, Dahleh M (2015) Optimal management and sizing of energy storage under dynamic pricing for the efficient integration of renewable energy. *IEEE Trans. Power Systems* 30(3): 1164–1181.

Hough PD, Kolda TG, Torczon VJ (2001) Asynchronous parallel pattern search for nonlinear optimization. *SIAM J. Sci. Comput.* 23(1):134–156.

Johnson SA, Stedinger JR, Shoemaker CA, Li Y, Tejada-Guibert JA (1993) Numerical solution of continuous-state dynamic programs using linear and spline interpolation. *Oper. Res.* 41(3): 484–500.

Jones DR (2001) A taxonomy of global optimization methods based on response surfaces. *J. Global Optim.* 21(4):345–383.

Kakade SM (2002) A natural policy gradient. Dietterich TG, Becker S, Ghahramami Z, eds. *Proc. 14th Internat. Conf. Neural Inform. Processing Systems: Natural Synthentic* (MIT Press, Cambridge, MA), 1531–1538.

Keller PW, Mannor S, Precup D (2006) Automatic basis function construction for approximate dynamic programming and reinforcement learning. *Proc. 23rd Internat. Conf. Machine Learn.* (ACM, New York), 449–456.

Kim JH, Powell WB (2011) Optimal energy commitments with storage and intermittent supply. *Oper. Res.* 59(6):1347–1360.

Kober JR, Peters JR (2009) Policy search for motor primitives in robotics. Koller D, Schuurmans D, Bengio Y, Bottou L, eds. *Adv. Neural Inform. Processing Systems 21* (Curran Associates, Inc., Red Hook, NY), 849–856.

Kolda TG (2005) Revisiting asynchronous parallel pattern search for nonlinear optimization. *SIAM J. Optim.* 16(2):563–586.

Kormushev P, Caldwell DG (2012) Direct policy search reinforcement learning based on particle filtering. *10th Eur. Workshop Reinforcement Learn. (EWRL 2012), Edinburgh, UK.*

Kuhn D, Wiesemann W, Georghiou A (2011) Primal and dual linear decision rules in stochastic and robust optimization. *Math. Programming* 130(1):177–209.

Lai G, Margot F, Secomandi N (2010) An approximate dynamic programming approach to benchmark practice-based heuristics for natural gas storage valuation. *Oper. Res.* 58(3):564–582.

Lewis RM, Torczon V, Trosset MW (1998) Why pattern search works. *Optima* (59):1–7.

Lewis RM, Torczon V, Trosset MW (2000) Direct search methods: Then and now. *J. Comput. Appl. Math.* 124(1):191–207.

MacKay DJ (2009) *Sustainable Energy-Without the Hot Air*, 1st ed. (UIT Cambridge Ltd., Cambridge, UK).

Mannor S, Rubinstein R, Gat Y (2003) The cross entropy method for fast policy search. *Proc. 20th Internat. Conf. Machine Learn. (ICML-2003), Washington, DC*, 512–519.

Moazeni S, Coleman TF, Li Y (2016) Smoothing and parametric rules for stochastic mean-CVAR optimal execution strategy. *Ann. Oper. Res.* 237(1):99–120.

Moazeni S, Powell WB, Hajimiragha AH (2015) Mean-conditional value-at-risk optimal energy storage operation in the presence of transaction costs. *IEEE Trans. Power Systems* 30(3): 1222–1232.

Munos R (2003) Error bounds for approximate policy iteration. *Proc. 20th Internat. Conf. Machine Learn. (ICML-2003), Washington, DC*, 560–567.

Ng AY, Jordan M (2000) PEGASUS: A policy search method for large MDPs and POMDPs. *Proc. 16th Conf. Uncertainty Artificial Intelligence* (Morgan Kaufmann Publishers Inc., San Francisco), 406–415.

Nocedal J, Wright S (2006) *Numerical Optimization*, 2nd ed. (Springer, New York).

Peters J, Vijayakumar S, Schaal S (2005) Natural actor-critic. Gama J, Camacho R, Brazdil PB, Jorge AM, Torgo L, eds. *Machine Learning: ECML 2005: 16th Eur. Conf. Maching Learn.*, Lecture Notes in Computer Science, Vol. 3720 (Springer, Berlin), 280–291.

Powell WB (2011) *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed. (John Wiley & Sons, New York).

Riedel F (2004) Dynamic coherent risk measures. *Stochastic Processes Their Appl.* 112(2):185–200.

Rios LM, Sahinidis NV (2013) Derivative-free optimization: A review of algorithms and comparison of software implementations. *J. Global Optim.* 56(3):1247–1293.

Rockafellar RT, Uryasev S (2000) Optimization of conditional value-at-risk. *J. Risk* 2(3):21–41.

Rudloff B, Street A, Valladao DM (2014) Time consistency and risk averse dynamic decision models: Definition, interpretation and practical consequences. *Eur. J. Oper. Res.* 234(3):743–750.

Scott WR, Frazier P, Powell WB (2011) The correlated knowledge gradient for simulation optimization of continuous parameters using Gaussian process regression. *SIAM J. Optim.* 21(3): 996–1026.

Scott WR, Powell WB, Moazeni S (2014) Least squares policy iteration with instrumental variables vs. direct policy search: Comparison against optimal benchmarks using energy storage. http://arxiv.org/pdf/1401.0843v1.pdf.

Shafi A, Carpenter B, Baker M (2009) Nested parallelism for multi-core HPC systems using java. *J. Parallel Distributed Comput.* 69(6):532–545.

Shapiro A (2012) Time consistency of dynamic risk measures. *Oper. Res. Lett.* 40(6):436–439.

Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M (2014) Deterministic policy gradient algorithms. *Proc. 31st Internat. Conf. Machine Learn., Beijing*, I-387–I-395.

Strens MJA, Moore AW (2003) Policy search using paired comparisons. *J. Machine Learn. Res.* 3(3):921–950.

Sutton R, Barto A (1998) *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA).

Sutton RS, McAllester D, Singh S, Mansour Y (2000) Policy gradient methods for reinforcement learning with function

approximation. *Advances in Neural Information Processing Systems*, Vol. 12 (MIT Press, Cambridge, MA), 1057–1063.

Torczon V (1997) On the convergence of pattern search algorithms. *SIAM J. Optim.* 7(1):1–25.

Tsitsiklis JN, Roy BV (1996) Feature-based methods for large scale dynamic programming. *Machine Learn.* 22(1):59–94.

Yamakawa E, Fukushima M (1996) A block-parallel conjugate gradient method for separable quadratic programming problems. *J. Oper. Res. Soc. Japan* 39(3):407–427.

Yu H, Bertsekas DP (2009) Basis function adaptation methods for cost approximation in MDP. *IEEE Sympos. Adaptive Dynam. Programming Reinforcement Learn., Nashville, TN*, 74–81.

Yu H, Bertsekas DP (2010) Error bounds for approximations from projected linear equations. *Math. Oper. Res.* 35(2):306–329.

Zhou Y, Scheller-Wolf A, Secomandi N, Smith S (2014) Managing wind-based electricity generation in the presence of storage and transmission capacity. Tepper Working Paper 2011-E36 1–38, Carnegie Mellon University, Pittsburgh, PA.