Optimization of a Large-Scale Water Reservoir Network by Stochastic Dynamic Programming with Efficient State Space Discretization

COSMOS Technical Report 04-04

Cristiano Cervellera (corresponding author)
Institute of Intelligent Systems for Automation - ISSIA-CNR National Research Council of Italy, Via De Marini 6, 16149 Genova, Italy. Email: cervellera@ge.issia.cnr.it

Victoria C. P. Chen
Dept. of Industrial & Manufacturing Systems Engineering - The University of Texas at Arlington, Campus Box 19017 Arlington, TX 76019-0017. Email: vchen@uta.edu

Aihong Wen
Dept. of Industrial & Manufacturing Systems Engineering - The University of Texas at Arlington, Campus Box 19017 Arlington, TX 76019-0017. Email: axw0932@omega.uta.edu
Abstract
A numerical solution to a 30-dimensional water reservoir network optimization problem, based on stochastic dynamic programming, is presented. In such problems the amount of water to be released from each reservoir is chosen to minimize a nonlinear cost (or maximize benefit) function while satisfying proper constraints. Experimental results show how dimensionality issues, given by the large number of basins and realistic modeling of the stochastic inflows, can be mitigated by employing neural approximators for the value functions, and efficient discretizations of the state space, such as orthogonal arrays, Latin hypercube designs and low-discrepancy sequences.

Keywords
Dynamic programming, Large-scale optimization, Applied probability, Neural networks, Natural resources
1 Introduction

Optimal operation of water reservoir networks has been studied extensively in literature (e.g., [1, 12, 13, 21, 34]; Yakowitz [37] provides an excellent survey). Reservoir networks systems can be represented by graphs in which the nodes correspond to water basins and the links are characterized by interbasin transfers. In typical models, the inputs to the nodes are water released from upstream reservoirs and stochastic inflows from external sources (like rivers and rain), while the outputs correspond to the amount of water to be released during a given time period (e.g., a month).

The dynamics for the single basin can be modeled by a state equation where the amount of water at the beginning of period $t + 1$ reflects the flow balance between the water that enters (upstream releases and stochastic inflows) and the water that is released during period $t$. The amount of water to be released during a given time period from each reservoir is chosen to minimize some possibly nonlinear cost (or maximize benefit) function related to the releases (e.g., power generation), while satisfying proper constraints, e.g., maximum pumpage capacities or target water level in each basin at the beginning of each period $t$. Thus, optimal management of the reservoir network can be formulated as an optimization problem, in which the aim is to determine the quantity of water releases that minimize a total cost over a given horizon of $T$ time periods (e.g., a year). In the case of large-scale reservoir networks with several basins, nonlinearities and the presence of stochastic variables, the corresponding optimization problem becomes very complex.

This is particularly evident when we want to model the stochastic inflows accurately. In a realistic representation of inflow dynamics, the amount of (random) water flowing into the basins during a given period $t$ depends on the amounts of past periods. An example of realistic modeling, widely and successfully employed in the water resources literature [13, 30], is to consider autoregressive linear models of a given order $k$. To construct this model, the inflows of the past $k$ time periods must be included in the state vector, which consequently can become very large.

Stochastic dynamic programming (SDP) [3, 4, 28] is the most commonly used solution technique in the reservoir networks management literature [1, 12, 13, 20, 27, 37]. SDP is based on the definition, at each stage $t$, of a value function which quantifies the cost from that stage through the end of the time horizon. In this way, it is possible to transform the optimization problem into the recursive solution of a sequence of simpler optimization subproblems. It is well known that an exact solution to the SDP equations can be obtained only when system dynamics are linear and the cost (benefit) function is quadratic. In the general case we must seek approximate solutions, which are based on a state space discretization and an approximation of the value functions over the continuous space. Although for the deterministic case there exist efficient versions of dynamic programming, such as Differential DP [19], classical approaches for the stochastic case suffer from the curse of dimensionality phenomenon, which is an exponential growth of the computational and memory requirements as the dimension of the state vector increases. This is why only reservoir networks of limited size are considered in the literature, in the absence of restrictive hypotheses on the model. For example, in a recent paper [27], the largest example involved seven state variables. Therefore, for large-scale reservoir networks with accurate inflow dynamics, a very efficient version of SDP is still needed.

In this work we present approximate solutions to a 30-dimensional problem. The state space dimension arises from a test network with 10 reservoirs and inflows modeled by an
autoregressive system of order 2, where the cost and the constraints are nonlinear. To the best of the authors’ knowledge, this is the largest dimensional problem ever addressed in the reservoirs management literature by SDP techniques, at least without introducing restrictive hypotheses on the cost and/or the model.

For solving such a high-dimensional problem we utilize an approach based on efficient discretization of the state space and approximation of the value functions over the continuous state space by means of a flexible feedforward neural network. Other methods that employ neural networks for reservoir control problems can be found in literature. For instance, in [31], a 10-dimensional model where the cost function represents energy deficit in power generation is solved by an algorithm based on a two-phase neural network.

The approach presented here is based on the high-dimensional continuous-state SDP work of Chen [7, 8, 9]. The most commonly employed discretization in the literature, consists of a uniform grid of points over the state space, which is not efficient because it is subject to exponential growth. Referring to the area of statistical design of experiments, we consider more efficient discretizations. Specifically, orthogonal arrays (OA) [8], OA-based Latin hypercubes (OA-LH) [33], and number-theoretic methods based on Sobol’ [32] and Niederreiter-Xing [25, 26] low-discrepancy sequences. For what concerns low-discrepancy sequences in particular, it has been recently proven [6] that their use in the context of function learning leads to an almost linear sample complexity in the estimation of the best network inside the chosen family, which is in general one of the main sources of curse of dimensionality when value functions have to be approximated in SDP problems.

The paper is organized as follows. Section 2 contains a description of the 30-dimensional test reservoir network model. In Section 3 the method based on SDP is introduced and discussed. Section 4 is devoted to the actual solution of the test problem, where the relevant parameters and results are presented. Section 5 contains some concluding remarks.

2 The Model

The water reservoir network we consider consists of 10 basins, each one affected by stochastic inflows and controlled by means of water releases. The configuration of the network is depicted in Figure 1. The details concerning the model, defined in the following subsections, reflect typical situations that can be found in the reservoirs management literature.

2.1 Notation

We define

- \( \mathbf{w}_t^i \in \mathbb{R} \): amount of water in the \( i \)-th reservoir at the beginning of stage \( t, i = 1, \ldots, 10, t = 1, \ldots, T \); vector \( \mathbf{w}_t \in \mathbb{R}^{10} \).
- \( \mathbf{r}_t^i \in \mathbb{R} \): amount of water released from the \( i \)-th reservoir during stage \( t, i = 1, \ldots, 10, t = 1, \ldots, T \); vector \( \mathbf{r}_t \in \mathbb{R}^{10} \).
- \( \mathbf{e}_t^i \in \mathbb{R} \): stochastic net inflow into the \( i \)-th reservoir during stage \( t, i = 1, \ldots, 10, t = 1, \ldots, T \); vector \( \mathbf{e}_t \in \mathbb{R}^{10} \).
- \( z_t^i = e_{t-1}^i \) for \( t = 2, \ldots, T \); \( z_t^i = \hat{z}^i \) otherwise, where \( \hat{z}^i \) is given; vector \( \mathbf{z}_t \in \mathbb{R}^{10} \).
- \( y_t^i = e_{t-2}^i \) for \( t = 3, \ldots, T \); \( y_t^i = \hat{y}^i \) otherwise, where \( \hat{y}^i \) is given; vector \( \mathbf{y}_t \in \mathbb{R}^{10} \).
• $x_t = [w_t, z_t, y_t] \in \mathbb{R}^{30}$: the 30-dimensional state vector.
• $\xi_t \in \mathbb{R}$: independent standard normal distribution random variables; vector $\xi_t \in \mathbb{R}^{10}$.
• $\tilde{w}_t$: target water level for reservoir $i$ at the beginning of period $t$.
• $U^i$: the set of indexes corresponding to the reservoirs which release water into reservoir $i$.
• $W^i$: the maximum capacity of reservoir $i$.
• $R^i$: the maximum pumpage capability for reservoir $i$.

2.2 The stochastic inflows

The random vector $\epsilon_t$ is modeled through an autoregressive system of order 2, affected by a random correction $\xi_t$ that follows a standard normal distribution.

$$\epsilon_t^i = a_t^i \epsilon_{t-1}^i + b_t^i \epsilon_{t-2}^i + c_t^i + d_t \xi_t^i$$

The coefficients of the linear combinations ($a_t, b_t, c_t, d_t \in \mathbb{R}$) actually depend on $t$, so that it is possible to model proper inflows behaviour for different months. The actual values of the coefficients used for the model were based on the one-reservoir real model described in [13] and extended to our 10-reservoir network. In our example, we have chosen the values of the coefficients to be equal for reservoirs that are supposed to be “close” to each other (i.e., there are three “sets” of coefficients, corresponding to reservoirs 1-5, reservoirs 6-9 and reservoir 10, respectively), to model some correlation between inflows corresponding to similar locations. It is worth noting that more complex inflows dynamics that model greater correlation (e.g., where the autoregressive coefficients $a_t$ and $b_t$ do not form diagonal matrices) might be employed without affecting the complexity of the SDP method.

2.3 The state equation

The amount of water at stage $t + 1$ reflects the flow balance between the water that enters (upstream releases and stochastic inflows) and the water that is released. In order to deal with unexpected peaks of stochastic inflows, each reservoir has a floodway, so that the amount of water never exceeds the maximum value $W^i$. This is done to introduce a nonlinearity in the state equation of our test problem, and is consistent with actual real world examples (see, for instance, [14]). Therefore, the state equation for $w^i_t$ is

$$w_{t+1}^i = \min \left\{ w_t^i + \sum_{j \in U^i} r_j^i - r_t^i + \epsilon_t^i , W^i \right\}$$

For $z_t^i$ and $y_t^i$, we simply have

$$z_{t+1}^i = \epsilon_t^i$$
$$y_{t+1}^i = z_t^i$$
The three equations above can be grouped in the compact state equation
\[ x_{t+1} = f_t(x_t, r_t, \epsilon_t) \]
where the domain of the function \( f_t \) is a finite interval for \( w^i_t \) and \( r^i_t \) (\( 0 \leq w^i_t \leq W^i \) and \( 0 \leq r^i_t \leq R^i \)) and the whole real line for \( z^i_t, y^i_t \) and \( \epsilon^i_t \).

2.4 Constraints

We consider constraints on the water releases based on two different requirements

- each release is limited by a maximal pumpage capability;
- we adopt a “conservative” rule: the release can never exceed the amount at the beginning of the period plus the water that gets in from the upstream reservoirs. It is conservative because we do not consider the possible amount coming from the inflows.

Therefore, the constraints for the \( i \)-th control \( r^i_t \) are of the form

\[ 0 \leq r^i_t \leq \min \left\{ w^i_t + \sum_{j \in U^i} r^j_t, R^i \right\} \]

This also implies nonnegativity of \( w^i_{t+1} \).

2.5 Cost function

We consider the following objectives:

- keep the level of the water in the reservoirs, at the beginning of the new stage, as close as possible to a target value \( \tilde{w}^i_t \), smaller than the maximum capacity.
- minimize the cost (maximize benefit) represented by some possibly nonlinear and not convex generic function \( g \) of the water releases and/or water levels (e.g., power generation, irrigation, etc.)

In our example, we consider a nonlinear convex function \( g \) of the following form (for \( z \in [0, +\infty) \))

\[
\begin{align*}
g(z, \delta) &= \begin{cases} 
\frac{z^3}{3\delta^2} - \frac{z^4}{16\delta^3} & 0 \leq z \leq 2\delta \\
\frac{z^2}{\delta} & z > 2\delta
\end{cases}
\end{align*}
\]

Such a function models a benefit which becomes relevant for “large” values of the water releases, depending on a suitable parameter \( \delta \). Figure 2 depicts the behaviour of the function \( g \). Then, the total cost function at stage \( t \) can be written as

\[
h_t(x_t, r_t, \epsilon_t) = \sum_{i=1}^{10} |w^i_{t+1} - \tilde{w}^i_{t+1}| - \sum_{i=1}^{10} p^i g(r^i_t, \delta^i)
\]

\footnote{A list of different cost and benefit functions commonly employed in reservoir networks management can be found in [37].}
where $p^i \in \mathbb{R}$ for $i = 1, \ldots, 10$.

This cost function at stage $t$ is separable in all its variables, and the terms with absolute deviations are convex while the other terms are concave. However, it must be noted that the algorithm described in this work is not dependent on the state and the cost equations. Therefore, other formulations for $g$ or the penalty on the water levels $\tilde{w}^i$ could be employed, without changing the applicability of the proposed method\(^2\).

### 2.6 The optimization problem

We want each release $r_t$ to be a function of the current state vector (closed-loop control)

$$r_t = \mu_t(x_t)$$

Therefore, with the state equation, the constraints and the cost above defined, we can state the optimization problem as finding the optimal sequence $r^o = (\mu_1^o(x_1), \ldots, \mu_T^o(x_T))$ that solves

$$\min_{r_1, \ldots, r_T} E \left\{ \sum_{t=1}^{T} h_t(x_t, r_t, \epsilon_t) \right\}$$

subject to

$$x_1 = \tilde{x}$$

$$x_{t+1} = f_t(x_t, r_t, \epsilon_t)$$

$$0 \leq r^i_t \leq \min \left\{ w^i_t + \sum_{j \in U^i} r^j_t, R^i \right\}$$

where $\tilde{x}$ is a given initial state.

### 3 SDP solution to the optimization problem

The SDP approach for the solution of a finite-horizon optimization problem is based on the definition of a value function for a given time period, which corresponds to the optimal cost to operate the system from the current time period through the end of the time horizon. This value function is a function of the state variables, i.e., the optimal cost to operate the system depends on the states. Formally, the value function can be written recursively as

$$F_t(x_t) = \min_{r_t} E \left[ h_t(x_t, r_t, \epsilon_t) + F_{t+1}(x_{t+1}) \right]$$

subject to

$$x_{t+1} = f_t(x_t, r_t, \epsilon_t)$$

$$0 \leq r^i_t \leq \min \left\{ w^i_t + \sum_{j \in U^i} r^j_t, R^i \right\}$$

where the recursion is initialized by $F_T(x_T) \triangleq \min_{r_T} E h_T(x_T, r_T, \epsilon_T)$, which corresponds to $F_{T+1}(x_{T+1}) \triangleq 0$.

It is clear that the optimal cost of the problem (1) corresponds to $F_1(\tilde{x})$. If we obtain the value functions for all the time periods, then given the initial state of the system, we can track the optimal solution through the value functions. Thus, a DP solution is fully specified by solving for the value functions. Further details can be found in [4, 28].

\(^2\)For instance, in the particular case of benefit from power generation, $g$ might be a function of the water head variations too.
Unfortunately, it is well known that the value function equation can be analytically solved only when the state equation is linear and the cost function is quadratic (LQ hypotheses), which is not true in the case of typical reservoir management problems. Furthermore, the analytical solution is not valid when inequality constraints are present. Therefore, as we deal with continuous states, a numerical SDP solution requires a computationally-tractable approximation method for the value functions. In particular, when solving for $F_t$, we must be able to estimate the value of $F_{t+1}$ at any given point $x_{t+1}$.

This leads to the following algorithm

1. For each time period $t = T, \ldots, 1$: Choose a set of $N$ points $P_t$ in the state space for $x_t$.
2. In the last period $T$:
   (a) For each point $x_T \in P_T$, solve
   $$F_T(x_T) \triangleq \min_{r_T, \epsilon_T} E[h_T(x_T, r_T, \epsilon_T)].$$
   (b) Then approximate $F_T(x_T)$ by $\hat{F}_T(x_T)$, over the state space for $x_T$, using the function outputs obtained for $F_T$ from step 2a.
3. In each period $t = T - 1, \ldots, 1$:
   (a) For each point $x_t \in P_t$, solve
   $$\tilde{F}_t(x_t) = \min_{r_t, \epsilon_t} E[h_t(x_t, r_t, \epsilon_t) + \hat{F}_{t+1}(f_t(x_t, r_t, \epsilon_t))].$$
   (b) Then approximate $\tilde{F}_t(x_t)$ with $\hat{F}_t(x_t)$, over the state space for $x_t$, as in step 2b.

The algorithm is suboptimal, in the sense that $\tilde{F}_t(x_t)$ is an approximate value of the true value function. The suboptimality level is given by (i) how accurately the minimum value function is computed, (ii) how accurate the estimation of the expected value is with respect to $\epsilon_t$ (typically, by averaging over a finite number of realizations) and (iii) how close the approximation $\hat{F}_t(x_t)$ is to the true value function.

The minimization problem in steps 2a and 3a must be solved $N$ times, and each solution requires substantial computational effort. Therefore, the choice of a good set of points $P_t$, in the sense that its size does not need to grow too much with the dimension of the state space in order to ensure a desired accuracy of approximation, is crucial for acceptable computational requirements. This is particularly true for very high dimensional contexts such as our 30-dimensional reservoirs problem.

Since we have no prior knowledge of the form of the value function, it is reasonable to choose sets which cover the state space uniformly. The most widely used technique in the DP literature consists of an uniform grid over the state space, which grows exponentially in the number of points with the dimension of the state vector (curse of dimensionality).

Recently, more advanced discretization techniques have been proposed in literature. In [29] it is proven that Monte Carlo sampling can break the curse of dimensionality of
DP, but only for particular cases, and when the set of possible decisions is finite (which is not the case of water releases in reservoirs operation). On the other hand, deterministic experimental designs are receiving a great deal of attention due to their successful application in many fields [11, 18, 22, 23, 24]. For SDP, OA experimental designs have permitted a solution to a nine-dimensional inventory forecasting problem [9]. The OA, OA-LH, and number-theoretic designs we utilize here are discussed further in Section 4.2.

For the choice of approximator to estimate the value functions, the requirement for a computationally-tractable solution is to use structures that can approximate high-dimensional nonlinear functions with sufficient accuracy and without excessive computational complexity. In particular, the complexity of the approximator must not grow exponentially in order to obtain a certain level of accuracy as the dimension of the input vector grows (which can be seen as another form of curse of dimensionality).

In this paper we use feedforward neural networks [16], due to their well-known theoretical properties as universal approximators [2] and their success in a wide range of different application fields, including DP [5].

4 Solving the 30-dimensional Model

4.1 Feedforward Neural Networks

As known, the basic architecture of feedforward neural networks is an interconnection of neural nodes that are organized in layers. The input layer consists of \( n \) nodes, one for each input variable, and the output layer consists of one node for the output variable. In between there are hidden layers which induce flexibility into the modeling. Activation functions define transformations between layers, the simplest one being a linear function. A comprehensive description of various forms may be found in [16].

In our reservoirs management SDP application, we utilized a feedforward one-hidden-layer NN (FF1NN) with a “hyperbolic tangent” activation function: 

\[
\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.
\]

Define \( n \) as the number of input variables, \( Q \) as the number of hidden nodes, \( \beta_{kq} \) as weights linking input nodes \( k \) to hidden nodes \( q \), \( \alpha_q \) as weights linking hidden nodes \( q \) to the output node, and \( \theta_q, \gamma \in \mathbb{R} \) are called “bias” nodes.

Then our FF1NN model form for the approximation of the \( t \)-th value function is

\[
\hat{F}_t(x; \alpha, \beta, \theta, \gamma) = \sum_{q=1}^{Q} \alpha_q Z_q + \gamma,
\]

where for each hidden node \( q \)

\[
Z_q = \text{Tanh} \left( \sum_{i=1}^{n} \beta_{iq} x_i + \theta_q \right).
\]

For our tests, the training of the parameters \( \beta_{iq}, \alpha_q, \theta_q, \gamma \) was performed by minimizing a mean squared error (MSE) between the output of the network and the true value functions in the \( N \) points of the set \( P_t \) (the set of the input-output pairs used for the training takes the name of training set)

\[
\text{MSE}(\hat{F}_t) = \sum_{j=1}^{N} [F_t(x_{t,j}) - \hat{F}_t(x_{t,j}; \alpha, \beta, \theta, \gamma)]^2,
\]
where $x_{t,j}$ indicates the $j$-th point of the set $P_t$. The nonlinear optimization technique used to conduct the minimization was the Levenberg-Marquardt method (one of the best training algorithms for feedforward neural networks, in terms of rate of convergence, first introduced in [15]), which is an approximation of the classic second-order (i.e., it makes use of the Hessian matrix) Newton method.

For what concerns the choice of the number $Q$ of nodes in the hidden layer, existing theoretical bounds on the accuracy of the approximation are non-constructive or too loose to be useful in practice. Thus, it is not possible to define “a priori” the optimal number of hidden nodes, since it depends on the number of samples available and on the smoothness of the function we want to approximate. Too few nodes would result in a network with poor approximation properties, while too many would “overfit” the training data (i.e., the network merely “memorizes” the data at the expense of generalization capability). Therefore, a tradeoff between these two possibilities is needed. There are many rules of thumb and methods for model selection and validation in the literature, as well as methods to avoid or reduce overfitting [16], but the choice of a good architecture usually requires several modeling attempts and is always very dependent on the particular application.

## 4.2 State Space Discretization

For the solution of our 30-dimensional reservoirs management test problem, efficient discretization of the state space was critical for computational practicality. In a statistical perspective, state space discretization is equivalent to experimental design. The traditional full grid of points is equivalent a full factorial statistical design of experiment. Since the full factorial is considered by statisticians to be too large in practice for high dimensions, a variety of more efficient experimental designs have emerged (see [10] for a review). In this paper, we tested four types of designs: (i) orthogonal array (OA) of strength two, (ii) OA-based Latin hypercubes (OA-LH), (iii) low-discrepancy Sobol’ (SB) sequences, and (iv) low-discrepancy Niederreiter-Xing (NX) sequences. For each design type, we constructed discretizations with $N = 961$ and $N = 1849$ points, for a total of 8 different discretizations.

First consider a design with $p$ levels in each of $n$ dimensions. A full grid would consist of $N = p^n$ points. An OA design of strength $d$ is a fractional factorial design with the property that if we look at any $d$ of the $n$ dimensions, then each of the $p^d$ possible combinations of levels of these $d$ variables occurs equally often, say $\lambda$ times, in the design. Thus, an OA design has $N = \lambda p^d$. In our case we chose $\lambda = 1$ and strength $d = 2$. Spatially, when the design points of an OA of strength two with $\lambda = 1$ are projected onto any two-dimensional subspace, each point of the two-dimensional full factorial with $p$ levels in each dimension will be represented exactly once. Specifically, we used $p = 31$ to generate $N = 961$ points and $p = 43$ to generate $N = 1849$ points.

A Latin hypercube (LH) design is mathematically equivalent to an OA of strength one (although such an OA is technically not an orthogonal design). When the points of an LH are projected onto any single dimension, all $p$ levels will be represented exactly once. Thus, we can consider $N = p$ for an LH. High correlations between dimensions are unfortunately possible in an LH design, and hybrid OA-LH designs attempt to overcome this drawback. An OA-LH design takes the points of an OA and utilizes a mapping on the $p$ levels to convert the design into an LH. For our OA-LH designs, we utilized the 961-point and 1849-point OA design previously mentioned.
A number-theoretic method, a.k.a., quasi-Monte Carlo method, uses number theory and numerical analysis to generate a point set that is uniformly-spaced. The general study of uniformly distributed sequences was initiated in 1916 by Herman Weyl [35, 36], who defined the notion of “discrepancy” to quantify the quality of uniformity of a finite point set. Current research seeks to construct “low-discrepancy” sequences. Define an elementary interval in base \( q \) over \([0, 1)^s\) as an \( s \)-dimensional subrectangle with component intervals of the form:

\[
[a_j, b_j) = \left[ \frac{\alpha_j}{q^{l_j}}, \frac{\alpha_j + 1}{q^{l_j}} \right)
\]

for integers \( l_j \geq 0 \) and \( 0 \leq \alpha_j < q^{l_j} \). A \((t, m, s)\)-net in base \( q \) on \([0, 1)^s\) consists of \( q^m \) points such that every elementary interval in base \( q \) of volume \( q^{t-m} \) contains exactly \( q^t \) points. A \((t, s)\)-sequence in base \( q \) is an infinite sequence of points \( \{X_i\} \) such that for all integers \( k \geq 0 \) and \( m \geq t \), the finite sequence \( \{X_{kq^m+1}, \ldots, X_{(k+1)q^m+1}\} \) is a \((t, m, s)\)-net in base \( q \). Sobol’ sequences are \((t, s)\)-sequences in base 2 (C++ code obtained from \( \text{http://ldsequences.sourceforge.net/} \)), and Niederreiter-Xing sequences construct \((t, s)\)-sequences in base \( q \) (generators obtained from \( \text{http://www.dismat.oeaw.ac.at/pirs/niedxing.html} \)).

The use of low-discrepancy sequences in function learning contexts (as is the value function approximation problem in the SDP algorithm) has been studied in [6], where it is proven that such discretization schemes allow an almost linear sample complexity for the estimation error, i.e., the error between the “best” neural network for a given number of hidden nodes and the actual network obtained after the training decreases almost linearly with \( N \).

### 4.3 Parameters of the SDP solutions

The 30-dimensional reservoirs management SDP problem was solved over three time periods \( (T = 3) \). Conceptually, solving for longer horizons does not add difficulties to the SDP problem: as discussed in the previous sections, the aim of the proposed method is to address the curse of dimensionality that depends only on the size of the state vector, and not on the number of time stages (in fact, the computational requirements are just linear in the number of time periods \( T \)). What is to be expected is at each stage the error in the approximation of the optimal value functions “propagates” backwards. Unfortunately, establishing a “global” bound for such error is possible only when the DP problem satisfies specific assumptions. For example, there is the case of discounted problems, where in the DP equations the value function is weighted by a parameter \( 0 < \mu < 1 \) (see, e.g., [29]). In the general case, we must assume that the approximation errors sum through the stages.

Table 1 and 2 contain the most relevant numerical quantities for the various reservoirs. Table 7 contains the coefficients of the autoregressive model for the inflows dynamics corresponding to the first three time periods. The target levels \( \tilde{w}_t \) are constant for each \( t \), with values indicated in the tables. The considered initial point \( \tilde{x} \) corresponds to such target levels for the various reservoirs. The expectation of the value functions in equation (2) taken over the 10-dimensional random vector \( \xi_t \) were estimated using an average over a finite number of realizations from a standard normal distribution. Different numbers of realizations were tested\(^3\), and in the end 10 realizations were chosen as a compromise between accuracy and computational effort. Using more than 10 realizations increased

---

\(^3\)In particular, costs \( F_t(x_t) \) were computed for a set of test points \( x_t \) at various stages, averaging over increasing numbers of realizations.
the computational time without leading to significantly different costs\textsuperscript{4}. In any case, while more accurate results might be achieved with more realizations, the approximation of the expected value is not a key aspect of this paper since it is not affected by curse of dimensionality (see, e.g., Hoeffding’s inequality \cite{17}).

Many solutions were obtained by testing different values of $Q$. Depending on the size and kind of design employed, FF1NNs with $Q = 10$ and $Q = 15$ hidden nodes gave the best results. Both the minimizations for the computation of the value function in a given point and the training of the FF1NN approximations were implemented in MATLAB, using the “Optimization” and “Neural Network” toolboxes\textsuperscript{5}. Computing a value function approximation for a single time period $t$ (which corresponds to (i) solving the minimization for the $N$ points of $P_t$ and (ii) training of the FF1NN) required approximately 2 hours and 45 minutes for $N = 961$ points and 10 neural units on a Pentium IV 1.60 GHz machine with 256Mb RAM.

4.4 Results

Since there are no analytical nor approximate solutions for the 30-dimensional problem to which we can compare the SDP method described here, the evaluation of the results is performed on the basis of many different solutions obtained with the SDP method itself. In our case, one SDP solution is defined by the discretization scheme employed, the number of neural units $Q$ and number of points $N$ of the discretization. Such a procedure has already been used in \cite{9} to evaluate the results of the SDP method applied to a nine-dimensional inventory forecasting problem.

Simulations using the “reoptimizing policy,” based on the forward solution of the value function equations (as described in \cite{7}), were conducted to find the optimal water releases for a given sequence of random inflows and test the SDP solutions. This represents the “on-line” forward phase of the optimization procedure: at the beginning of each stage, optimal releases for the actual value of the water levels are computed by equation (2) using the neural approximations obtained “off-line.” Then, such optimal releases are fed into the state equation, which provides the water levels for the next stage according to the actual random inflows, and the new releases can be computed in the same way. Starting from $\tilde{x}$, a simulation of 100 “on-line” random inflows sequences was conducted for each SDP solution. For a given solution (i.e., given $Q$, $N$ and discretization scheme), the simulation output provided the mean costs over the 100 sequences. In order to compare the different solutions, the smallest cost achieved among the set of all the various SDP solutions for each the 100 random inflows sequences was used as the “true” cost for that particular sequence. A total of 8 different solutions have been considered for the evaluation of the results, each one providing the lowest mean cost for each of the 8 combinations of design scheme and number of points $N$. The average “true” mean cost over the 100 sequences (i.e., the average obtained by taking the best cost for each sequence) was approximately $-282$. This is the mean cost to which the mean costs of the single SDP solutions have been compared.

Table 3 summarizes, for the aforementioned 8 solutions, the mean costs (over the 100 inflows), number of neural nodes $Q$, and % average absolute error with respect to the “true” mean cost over the 100 sequences.

\textsuperscript{4}For instance, using 100 realizations instead of 10 led to costs that were averagely only 1.5-2% different.

\textsuperscript{5}In particular, the function ‘fmincon’, based on a quasi-Newton algorithm, has been employed for the minimizations.
mean cost. Boxplots in Figure 3 display the distributions of the absolute errors for each solution with respect to the 100 random inflows sequences. The box marks the 25th and 75th percentiles of the distribution while the line inside marks the median value. Outlier points are seen beyond the lines extending from the box. The best solutions will have small boxes (low variance) that are close to zero (low absolute error). The worst solutions are those of the 961-point OA and OA-LH, the former because of higher absolute error and the latter because of larger variance. The remaining designs are somewhat equivalent; however, the 1849-point Sobol’ sequence is seen as providing the best solution.

Optimal trajectories for two different solutions, namely the 961-point OA design (OA-961) and the 1849-point Sobol’ sequence (SB-1849), were recorded for the two random inflow sequences shown in Table 4. Tables 5 and 6 compare the two solutions. It can be seen that the target water levels are generally attained at each stage, even for the “worst” solution in absolute error.

However, how close the levels are to the target levels depends on how much the other contribution to the cost (i.e., “benefit” part) is weighted through the coefficients $p^i$.

5 Conclusions

A computationally-tractable SDP method for the optimal management of large-scale water reservoir networks was presented. The use of innovative discretization schemes, such as OAs and $(t, m, s)$-nets, allows to deal effectively with the curse of dimensionality problems, arising from the number of basins and realistic modeling of the random inflows. Four different design types have been tested and compared in a 30-dimensional test problem. The employment of such discretization schemes has allowed the solution of a problem which would have been unsolvable by classic DP techniques. The great interest currently devoted to deterministic designs by the scientific community leads one to believe that a further understanding and development of these techniques, together with their application in the SDP framework, can provide new interesting opportunities for solving even larger and more difficult problems.

Acknowledgements

This work was partially supported by NSF Grants #INT 0098009 and #DMI 0100123 and a Technology for Sustainable Environment (TSE) grant under the U. S. Environmental Protection Agency’s Science to Achieve Results (STAR) program (Contract #R-82820701-0).

References


<table>
<thead>
<tr>
<th>Res. 1</th>
<th>Res. 2</th>
<th>Res. 3</th>
<th>Res. 4</th>
<th>Res. 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Water Level$^*$</td>
<td>200</td>
<td>250</td>
<td>260</td>
<td>270</td>
</tr>
<tr>
<td>Maximum Reservoir Capacity$^*$</td>
<td>433</td>
<td>420</td>
<td>460</td>
<td>440</td>
</tr>
<tr>
<td>Maximum Pumpage Capability$^*$</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
<tr>
<td>Coefficient $p$ of the benefit</td>
<td>0.15</td>
<td>0.12</td>
<td>0.11</td>
<td>0.13</td>
</tr>
<tr>
<td>Parameter $\delta$ of the benefit</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 1: Relevant numerical quantities for reservoirs 1-5 ($^*$ in $10^5$ cubic meters)

<table>
<thead>
<tr>
<th>Res. 6</th>
<th>Res. 7</th>
<th>Res. 8</th>
<th>Res. 9</th>
<th>Res. 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Water Level$^*$</td>
<td>420</td>
<td>200</td>
<td>500</td>
<td>180</td>
</tr>
<tr>
<td>Maximum Reservoir Capacity$^*$</td>
<td>860</td>
<td>820</td>
<td>972</td>
<td>495</td>
</tr>
<tr>
<td>Maximum Pumpage Capability$^*$</td>
<td>190</td>
<td>160</td>
<td>300</td>
<td>70</td>
</tr>
<tr>
<td>Coefficient $p$ of the benefit</td>
<td>0.09</td>
<td>0.13</td>
<td>0.21</td>
<td>0.23</td>
</tr>
<tr>
<td>Parameter $\delta$ of the benefit</td>
<td>10</td>
<td>8</td>
<td>25</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 2: Relevant numerical quantities for reservoirs 6-10 ($^*$ in $10^5$ cubic meters)

<table>
<thead>
<tr>
<th>Design</th>
<th>Q</th>
<th>Mean Cost</th>
<th>% Avg. Abs. Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>OA-961</td>
<td>15</td>
<td>-263.3488</td>
<td>6.5</td>
</tr>
<tr>
<td>LH-961</td>
<td>15</td>
<td>-273.2001</td>
<td>3.0</td>
</tr>
<tr>
<td>SB-961</td>
<td>15</td>
<td>-276.5661</td>
<td>1.8</td>
</tr>
<tr>
<td>NX-961</td>
<td>15</td>
<td>-275.7479</td>
<td>2.1</td>
</tr>
<tr>
<td>OA-1849</td>
<td>15</td>
<td>-277.3321</td>
<td>1.5</td>
</tr>
<tr>
<td>LH-1849</td>
<td>10</td>
<td>-274.5260</td>
<td>2.5</td>
</tr>
<tr>
<td>SB-1849</td>
<td>10</td>
<td>-278.8036</td>
<td>1.0</td>
</tr>
<tr>
<td>NX-1849</td>
<td>10</td>
<td>-277.1163</td>
<td>1.6</td>
</tr>
</tbody>
</table>

Table 3: Results of the best solutions for each design kind.
<table>
<thead>
<tr>
<th></th>
<th>Sequence 1</th>
<th></th>
<th>Sequence 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$t=1$</td>
<td>$t=2$</td>
<td>$t=3$</td>
</tr>
<tr>
<td>Res. 1</td>
<td>24.0616</td>
<td>29.9666</td>
<td>24.2993</td>
</tr>
<tr>
<td>Res. 3</td>
<td>22.4524</td>
<td>21.9788</td>
<td>26.7324</td>
</tr>
<tr>
<td>Res. 4</td>
<td>26.7606</td>
<td>27.1361</td>
<td>22.6921</td>
</tr>
<tr>
<td>Res. 5</td>
<td>22.3314</td>
<td>26.1024</td>
<td>18.3354</td>
</tr>
<tr>
<td>Res. 6</td>
<td>10.7030</td>
<td>18.2211</td>
<td>12.1199</td>
</tr>
<tr>
<td>Res. 7</td>
<td>11.3328</td>
<td>18.5393</td>
<td>10.9239</td>
</tr>
<tr>
<td>Res. 9</td>
<td>13.8086</td>
<td>17.3705</td>
<td>11.0396</td>
</tr>
<tr>
<td>Res. 10</td>
<td>9.1711</td>
<td>10.2328</td>
<td>5.5590</td>
</tr>
</tbody>
</table>

Table 4: Two test random inflows sequences.

<table>
<thead>
<tr>
<th></th>
<th>Trajectory OA-961</th>
<th></th>
<th>Trajectory SB-1849</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$t=2$</td>
<td>$t=3$</td>
<td>final</td>
</tr>
<tr>
<td>Res. 1</td>
<td>193.9817</td>
<td>202.7601</td>
<td>197.5512</td>
</tr>
<tr>
<td>Res. 2</td>
<td>248.3936</td>
<td>249.1060</td>
<td>243.0062</td>
</tr>
<tr>
<td>Res. 3</td>
<td>253.9718</td>
<td>254.8554</td>
<td>262.3820</td>
</tr>
<tr>
<td>Res. 4</td>
<td>267.4587</td>
<td>269.2587</td>
<td>265.2953</td>
</tr>
<tr>
<td>Res. 5</td>
<td>214.2049</td>
<td>218.8262</td>
<td>211.5961</td>
</tr>
<tr>
<td>Res. 6</td>
<td>417.1801</td>
<td>420.2837</td>
<td>421.3650</td>
</tr>
<tr>
<td>Res. 7</td>
<td>197.2793</td>
<td>200.5463</td>
<td>200.0649</td>
</tr>
<tr>
<td>Res. 8</td>
<td>499.1866</td>
<td>501.3877</td>
<td>498.5197</td>
</tr>
<tr>
<td>Res. 9</td>
<td>183.2954</td>
<td>180.4634</td>
<td>180.5232</td>
</tr>
</tbody>
</table>

Table 5: Optimal trajectories for sequence 1.
<table>
<thead>
<tr>
<th>Trajectory OA-961</th>
<th>Trajectory SB-1849</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t = 2$</td>
<td>$t = 3$</td>
</tr>
<tr>
<td>Res. 1</td>
<td>195.0115</td>
</tr>
<tr>
<td>Res. 2</td>
<td>244.2780</td>
</tr>
<tr>
<td>Res. 3</td>
<td>261.4377</td>
</tr>
<tr>
<td>Res. 4</td>
<td>266.4080</td>
</tr>
<tr>
<td>Res. 5</td>
<td>214.4870</td>
</tr>
<tr>
<td>Res. 6</td>
<td>413.8135</td>
</tr>
<tr>
<td>Res. 7</td>
<td>199.0183</td>
</tr>
<tr>
<td>Res. 8</td>
<td>501.5083</td>
</tr>
<tr>
<td>Res. 9</td>
<td>185.2577</td>
</tr>
<tr>
<td>Res. 10</td>
<td>341.4655</td>
</tr>
</tbody>
</table>

Table 6: Optimal trajectories for sequence 2.

<table>
<thead>
<tr>
<th>Res. 1-5</th>
<th>Res. 6-9</th>
<th>Res. 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t = 1$</td>
<td>$t = 2$</td>
<td>$t = 3$</td>
</tr>
<tr>
<td>a</td>
<td>1.28</td>
<td>.90</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>23.9</td>
<td>15</td>
</tr>
<tr>
<td>d</td>
<td>41</td>
<td>27.6</td>
</tr>
</tbody>
</table>

Table 7: Coefficients of the autoregressive process for the random inflows.
Figure 1: Reservoir Network.
Figure 2: Behaviour of the benefit function.
Figure 3: (1) OA-961 (2) LH-961 (3) SB-961 (4) NX-961 (5) OA-1849 (6) LH-1849 (7) SB-1849 (8) NX-1849