Optimist: a Python library for Water System Optimal Operation and Analysis using SDDP

Luciano Raso  
*Delft University of Technology, lraso@tudelft.nl*

David Dorchies  
*IRSTEA Montpellier, david.dorchies@irstea.fr*

Jan Kwakkel  
*Delft University of Technology, j.h.kwakkel@tudelft.nl*

Pierre Olivier Malaterre  
*IRSTEA Montpellier, olivier.malaterre@irstea.fr*

Follow this and additional works at: https://scholarsarchive.byu.edu/iemssconference

Part of the [Civil Engineering Commons](https://scholarsarchive.byu.edu/civil), [Data Storage Systems Commons](https://scholarsarchive.byu.edu/datastorage), [Environmental Engineering Commons](https://scholarsarchive.byu.edu/environmental), [Hydraulic Engineering Commons](https://scholarsarchive.byu.edu/hydraulic), and the [Other Civil and Environmental Engineering Commons](https://scholarsarchive.byu.edu/othercivil)

---

https://scholarsarchive.byu.edu/iemssconference/2016/Stream-D/26

This Event is brought to you for free and open access by the Civil and Environmental Engineering at BYU ScholarsArchive. It has been accepted for inclusion in International Congress on Environmental Modelling and Software by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Optimist: a Python library for Water System Optimal Operation and Analysis using SDDP

Luciano Raso¹, David Dorchies², Jan Kwakkel¹, Pierre Olivier Malaterre²
1. Delft University of Technology, Department of Policy Analysis (l.raso@tudelft.nl, j.h.kwakkel@tudelft.nl)
2. IRSTEA Montpellier, UMR G-EAU (david.dorchies@irstea.fr, pierre-olivier.malaterre@irstea.fr)

Abstract:
Stochastic Dual Dynamic Programming (SDDP) is valuable tool in water management, employed for operational water management (i.e. suggesting effective release rules), and cost-benefit analysis evaluations. SDDP solves a multistage stochastic programming problem when uncertainty is a Markov process, and the system model is convex. SDDP can handle complex interconnected problem. Despite its potential, SDDP use is limited to few specialists. We present Optimist, a python library for setting up an SDDP problem from components, i.e. elements and objectives, and their relationships. Elements presently developed are: streamflow input, reservoir, release, and river-reach. Objectives can be linear, used for energy production, and threshold, used for flood protection and environmental flow. Optimist largely simplifies the setting up a SDDP problem, and therefore it is dedicated to water resources management experts who want to use SDDP for their problems. Optimist is developed in Python using the Object Oriented Programming paradigm.

Keywords: Water Resources, Reservoir operation, Stochastic Dual Dynamic Programming, Cost Benefit analysis, Operational water management.

1 INTRODUCTION

In operational water management, a system regulator takes operational decisions to achieve some specific objectives. Operational decisions and other uncertain variables influence the system dynamics. As time goes on, uncertainty unfolds and new operational decisions are taken. In the simplest example, a reservoir operator must decide the daily release from the reservoir to achieve some defined objectives, such as energy production and flood prevention. The reservoir volume depends on both the release decision and the inflow to the reservoir. Future inflow is uncertain, but as time goes on, new information about the inflow arrives and new release decisions are taken.

Water system operation can be framed as a stochastic optimal control problem (Soncini-Sessa, Castelletti et al. 2007). Optimal control offers solutions to get the maximum benefit from a dynamic system. Applying optimal control to water systems has two valuable applications: i) in operational water management, optimal control can be employed for designing effective operational rules (Castelletti, Pianosi et al. 2008); ii) in water resources system analysis, optimal system operation offers optimal use of resources, supporting the development of economic analysis, such as cost-benefit analysis (Tilmant, Pinte et al. 2008).

Stochastic Dual Dynamic Programming (SDDP) is an advanced method for optimal operation of complex dynamic systems under uncertainty (Pereira and Pinto 1991, Shapiro 2011). SDDP belongs to the stochastic dynamic programming family, in which the multi-stage decision problem is split in many step-problems, and solved backward. SDDP can handle relatively large systems, and offers the possibility to model dynamics interactions of complex dynamic systems, such as multi-reservoir system, also including hydraulic routing. Despite its analytical potential, SDDP use within the water resources management domain is relatively limited. Setting a SDDP problem, in fact, is often beyond the know-how of water resources system analysts, whose expertise is related to the physical, economical, and social processes, or to the specific system at hand, rather than to dynamic programming methods.

To support water resource managers in using SDDP, we present Optimist. Optimist is a python library that enhances the application of SDDP for water resources system analysis by simplifying the
specification of the SDDP problem. In Optimist, users set a SDDP problem from pre-built water systems components.

2 STOCHASTIC DUAL DYNAMIC PROGRAMMING (SDDP)

In operational water management, decisions and information come iteratively one after another. Therefore, optimal operation can be framed as a multistage stochastic programming problem (Shapiro and Andrzej 2003). Stochastic Dynamic Programming (Bellman and Dreyfus 1966) solves a multistage stochastic programming when the problem is “separable”, i.e. it can be written as a combination of step-problems, and solved backwards. In each step-problem, the objective is the sum of present and future benefits.

SDDP is an extension of Stochastic Dynamic Programming. SDDP largely reduces the computational burden, under condition of convex step-problems. In SDDP, the present cost-to-go function is the sum of present benefit and expected future cost-to-go. Equation (1) represents the system objectives and Equation (2) the system dynamic and constraints.

\[
F_{t-1}(x_{t-1}) = \min f_t^T x_{t-1} + E_{x_t \in X_t} \left[ F_t(x_t) \right]
\]

With

\[
A_t x_t = B_{t-1} x_{t-1} + c_t  \\
A_t^{ineq} x_t \leq b_t^{ineq}
\]

In Equations (1) and (2), \(x_t \in \mathbb{R}^{N_t}\) are all problem variables at time \(t\), \(F_t\) is the cost-to-go function, \(f_t \in \mathbb{R}^{N_t}\) is the cost vector, \(A_t, B_{t-1}\) are matrices and \(c_t\) is a vector that defines the system dynamic. Both \(A_t\) and \(B_{t-1}\) are of dimension \(N_t \times N_{t-1}\), and \(c_t\) is of dimension \(N_{t-1}\), where \(N_t\) is the number of system states. \(A_t^{ineq}\) and \(b_t^{ineq}\) are constraints that apply to the system. In Equation (1-2), stochastic variables must be defined as Markov.

\[
p_t(x_t | x_{t-1}, \ldots, x_{t-n}) = p_t(x_t | x_{t-1})
\]

In Equation (3), \(p_t\) is the probability density function (pdf) of \(X_t\). Equation (3) states that \(x_{t-1}\) contains all the information to predict \(x_t\).

SDDP approximates the cost-to-go function \(F_t\) by its linear extrapolations, called cuts. The approximate cost-to-go function is built up in the forward and backward iterations, where new cuts are added at each iteration, resulting in better and better approximation. The iterations are repeated until convergence.

Setting up an SDDP problem requires defining \(f_t, A_t, B_{t-1}, c_t, A_t^{ineq}, b_t^{ineq}\), and \(p_t\), for all step-problems. Additionally, SDDP setting requires the following parameters: i) optimization horizon \(H\), ii) Time-step length, iii) backwards sampling size \(N\), iv) forward sampling size \(M\), and v) stopping criteria. Stopping criteria are defined by a maximum number of iterations and/or an accuracy level.

3 OPTIMIST

Optimist simplifies the process of SDDP problem setting in a water resources management context. The objective of Optimist is making SDDP available to water resources system analysts who need to solve a stochastic optimal control problem to answer their analytical questions.

In Optimist, the analyst sets up a System from problem components. Problem components are Elements and Objectives. Elements are physical entities present in the system, connected with one another in a directed network. Objectives are related to one or more element variables. Available components are described in detail at Section 3.1. Once components and their relation are defined in a system, Optimist builds up the SDDP problem by defining matrices, vectors and distributions as in Equations (1-3).
Optimist is implemented in Python, utilizing object oriented programming, which minimizes code redundancies and makes code maintenance and development very effective. This is especially welcome in research, in which code must be modified to meet new requirements that were not initially considered. Object oriented programming flexibility allows the developer to extend easily existing classes in order to meet the analyst need in answering specific questions.

3.1 Available Components

Components are Elements and Objectives. Presently available elements are: Streamflow, Reservoir, Release Decision, and River Reach. Presently available objectives are Energy, Flood Protection, and Ecology.

A streamflow is the discharge entering the system. The streamflow component has a time-series and a process model. The time-series is a Pandas time series containing the observed discharge at that station. The process model is a Markov model, identified on the time series. Streamflow process models presently available in Optimist are i) the Thomas-Fiering model (Loucks, Van Beek et al. 2005), and ii) the multiplicative log-normal linear model, introduced in Raso, Malaterre et al. (2016). See Appendix for detailed equations.

The reservoir is represented by the reservoir mass balance, as in Equation (7).

\[ v_t = v_{t-1} + \sum_{i=1}^{n_i} q_{i,t} - \sum_{j=1}^{n_j} q_{j,t} - e_t \]  \tag{4}

In Equation (7), \( q_{i,t} \) are the inflows, \( q_{j,t} \) are the outflows, and \( e_t \) the evaporation from the reservoir, at time \( t \). Inflows can be either uncontrolled streamflows or rivers. Outflows are release decisions. Reservoir volume is limited between an upper and lower boundary, as in Equation (9).

\[ v_{\text{min}} \leq v_t \leq v_{\text{max}} \]  \tag{5}

where \( v_{\text{min}} \) is set to 0, unless differently specified.

In Equation (6) evaporation is proportional to the reservoir surface, considering a linear surface-volume relation, as in Equation (8).

\[ e_t = e_\tau \cdot S_t \]  \tag{6}

\[ S_t = k_{S/v} \cdot v_t + S_0 \]

\( e_t \) is the seasonal evaporation rate, defined as time-series on a year.

A release is a decision of water diversion from a reservoir or from a river reach. Releases are limited between an upper and lower boundary, as in Equation (9).

\[ r_{\text{min}} \leq r_t \leq r_{\text{max}} \]  \tag{7}

where \( r_{\text{min}} \) is set to 0, unless differently specified. If there is no upper and/or lower boundary, \( r_{\text{min}} \) and/or \( r_{\text{max}} \) are set to infinite. Within the release class, \( \text{electric\_turbines} \) is a Boolean attribute that specifies if the release is used energy production.

A river reach is a connection between two elements, or a confluence of two or more inflows. Equation (11) defines the equation governing the river reach component.
In Equation (8), $q_{i,t}$ are one or more inputs and $q_{out,t}$ is the single output.

All objectives are weighted according to their relevance in the decision problem, or their cost and benefits. 

*Energy* is defined from the reservoirs having at least one release used for energy production. Energy is a linear combination of linear benefit of release through turbines and the volume of the relative reservoir. Weights depend on the system characteristics.

*Flood Protection* and *Ecology* are threshold-activated cost functions. Equation (9) present the threshold cost function.

$$\max(q_t - \overline{q}, 0)$$

Flood protection cost is zero when the discharge is below the flood threshold. Similarly, ecology cost is zero when the discharge is above the minimum environmental flow. In both cases exceeding or insufficient water flow is penalized proportionally to the distance from the threshold value.

### 3.3 Application

In this section we will introduce an application of Optimist to a hypothetical water system, using it for both operational water management and policy design purposes.

![Diagram of the hypothetical water system](image)

**Figure 1:** Schema of the hypothetical water system.

Figure 1 shows a scheme of the hypothetical water system under analysis. Reservoir $M$ has been recently constructed on River $M$ for energy production purposes. Reservoir $M$ volume is $v_{max} = 10e10$ m$^3$ and it has turbines capacity of $r_{max}=500$ m$^3$/s. The analyst task is designing the reservoir operational rules that maximize energy production, under the constraint of i) avoiding flood downstream and ii) maintain a minimum environmental flow. Flood happens when discharge is larger than $q_{flood} = 3000$ m$^3$/s, and the minimum environmental flow is set to $q_{env} = 70$ m$^3$/s.

The river water authority is also considering the construction a new reservoir also for energy production, called $D$, downstream of $M$. The water authority wants to analyse the economic feasibility of the project.
We show the details of the operational management design problem setting for Reservoir $M$. We first create the system. In the following, the Optimist code example has grey background.

```python
M_system = System('R')
```

We subsequently add system components, i.e. elements and objectives. All components must specify to which system they refer. We start defining the streamflow input at station $M$, also indicating the file containing its discharge data, in this case a csv file.

```python
M_streamflow = Streamflow('M', R_system, M_discharge.csv)
```

We create then Reservoir $M$, defining name, system, inflow elements, and capacity.

```python
M_reservoir = Reservoir('M', R_system, v_max, elements_up=M_streamflow)
```

We added the connection to the upstream streamflow, previously defined, directly at definition of the reservoir. Otherwise, the connection can be added later, by the method "connect", as in the following.

```python
R_system.connect(M_streamflow, M_reservoir)
```

The reservoir has two release decisions. Release through turbines and through spillages. Max release through turbines is limited to $r_{\text{max}}$. We introduce them in the system by defining their name, lower and upper boundary, whether they produce electricity, and the reservoir to which they belong.

```python
Release_Electr = Release('ReleaseElectricity', R_system, 0, r_max, electric_turbines=1, elements_up=M_reservoir)
Release_Spill = Release('ReleaseSpillage', R_system, 0, +inf, elements_up=M_reservoir)
```

Release and spillage discharges converge downstream of the reservoir. We use a river reach to add them up. In the river-reach definition, the inflow are defined as list of elements.

```python
Downstream = RiverReach('Res M Total Discharge', R_system, [Release_Electr, Release_Spill])
```

Once the physical model is set, we can add the system objectives. In this case: Energy production, flood protection, and minimum environmental flow. Energy production is defined by the electricity price only, whereas flood and environmental objectives require specifying the element and their threshold value.

```python
Energy_obj = Objective.set(60)
flood_obj = Threshold.flood_protection(10e3, Downstream, q_flood)
env_obj = Threshold.environmental_flow(100, Downstream, q_env)
```

We can now define the SDDP problem using the SDDP.set_problem method.

```python
SDDP.set_problem(R_system, H, T, M, N, stopping_criteria)
```

Where $H$, $T$, $M$, and $N$, and stopping_criteria are SDDP problem parameters, previously defined. SDDP problem solution can be used to get optimal operation rules for the reservoir.

Optimist can also be used to tackle the policy analysis problem. SDDP can be used to assess the value of an additional reservoir downstream, used for energy production, operated in combination with the reservoir upstream. Similar to the setting of the existing system, the analyst can add the downstream reservoir and the release decision.

### 3.3 Future Developments

To fully exploit the potential of SDDP, Optimist still requires some important features. Future work will focus on the following developments: i) a method to identify streamflow process from time-series data, ii) a method to solve the SDDP problem within Optimist. We also plan to make available new components, such as the irrigation district component, and a river reach model that includes simple hydraulic processes, such as delay and attenuation.
4 CONCLUSIONS

Stochastic Dual Dynamic Programming (SDDP) is a valuable tool in water resources system analysis. SDDP can be used for optimal operation under uncertainty of complex interconnected systems, and employed for both operational water management purpose (i.e. suggesting effective release rules) and cost-benefit analysis evaluations. Despite the SDDP potential, its application is limited to few specialists: setting a SDDP problem is a complex task, and water management analysts often resort to simpler simulation based methods. We presented Optimist, a Python library to set up a SDDP problem from water system components, i.e. elements and objectives. Elements presently available are: streamflow process, reservoir, release decision, and river reach. Objectives presently available are energy production, flood protection and environmental flow. We showed how to set up a SDDP problem using Optimist in a hypothetical system. The target user of Optimist are water resources system experts who may need SDDP for their analytical questions. Optimist can be used either for operational water management purposes, i.e. designing reservoir operational rules, or for economic analysis, for optimization guarantees the most efficient use of the resource. Optimist is a modular tool, and new components can be defined by users. We intend to increase Optimist functionalities by introducing important features and new components. Features that have a high priority to be developed in the future are: a method that identify the streamflow process model from data, and a method that solves the SDDP problem.

REFERENCES


APPENDIX

Equation (4) defines the Thomas-Fiering model.

\[
q_t = \bar{q}_t + \phi_1 \cdot \frac{\sigma_t}{\sigma_{t-1}} (q_{t-1} - \bar{q}_{t-1}) + \epsilon_t \cdot \frac{\sigma_t}{\sigma_{t-1}} \sqrt{1 - \phi_1^2}
\]

(10)

where \(t\) is the time index and \(\tau\) the time index in the year, \(\epsilon_t \sim N(0,1)\), \(\bar{q}_t\) is the climatic average discharge at \(\tau\), \(\sigma_t\) is the standard deviation of discharge at \(\tau\), and \(\phi_1\) the autocorrelation between \(q_t\) and \(q_{t-1}\). The Thomas-Fiering model is an additive streamflow process model classically used in the water management literature, corresponding to a standardized Periodic Autoregressive model of lag 1. Despite its simplicity, the Thomas-Fiering model runs the risk of producing negative streamflow values, which do not make sense from a physical point of view.

The streamflow process model presented in (Raso, Malaterre et al. 2016), defined in Equations (5-6), produces non-negative discharge values only.
where

$$\rho_{\tau,j} = \frac{\bar{q}_\tau}{\bar{q}_{\tau-i}}$$

$$\omega_{\tau,j} = \psi_{\tau,j} \cdot \bar{q}_\tau$$

$$\kappa_{\tau} = \bar{q}_\tau \cdot \left( 1 - \sum_{i=1}^{p} \phi_{\tau,j} - \sum_{j=1}^{q} \psi_{\tau,j} \right)$$

$$\xi_{\tau} \sim \ln N\left(0, \sigma_{\xi_{\tau}}\right)$$

Model in Equation (5-6) is a linear multiplicative periodic model, to be used in SDDP. This model identification is simple, and in some case it offers a better representation of residual pdf.