juliacon

HydroPowerModels.jl: A Julia/JuMP Package for Hydrothermal Economic Dispatch Optimization

Andrew W. Rosemberg[1, 2], Alexandre Street[1, 2], Joaquim D. Garcia[1, 2, 5], Thuener Silva[1, 3], Davi M. Valladão[1, 3], and Oscar Dowson[4]

[1]Laboratory of Applied Mathematical Programming and Statistics (LAMPS)
[2]Department of Electrical Engineering, PUC-RIO
[3]Department of Industrial Engineering, PUC-RIO
[4]Department of Industrial Engineering, Northwestern University, Evanston
[5]PSR-Inc

ABSTRACT

HydroPowerModels.jl is a Julia package for solving multistage, steady-state, hydro-dominated, power network optimization problems with stochastic dual dynamic programming (SDDP). Our state-of-the-art open source tool is flexible enough for practitioners in the electrical sector to test new ideas in an efficient way. This tool was made possible by the Julia language and the surrounding ecosystem of packages. We use *JuMP*, a package for mathematical programming modeling; *PowerModels.jl*, a JuMP-extension for power network optimization; and *SDDP.jl*, another extension that implements the SDDP algorithm.

Keywords

Julia, Optimization, Energy, Stochastic Programming, Optimal Power Flow

1. Introduction

The hydrothermal dispatch problem is important for the planning and operation of hydro-dominated electrical systems such as the Brazilian national grid. The objective is to coordinate generation, energy distribution, and hydro-storage management in order to minimize the cost of operation. In this context, the hydrothermal dispatch problem is a medium-term planning problem where uncertainties related to the hydrology (i.e., rainfall and other inflows) of the hydroelectric plants and consumer-demand have fundamental importance. Generation takes two main forms: i) hydro-generation with a low marginal cost, and ii) thermal-generation, with a high marginal cost. However, in hydro-dominated systems such as Brazil, there is often insufficient capacity of thermal generation to meet demand. Thus, the system-operator faces a trade-off between using water for cheap generation in the present, against conserving water for future periods of drought. Because of this trade-off, the hydrothermal dispatch problem is often modeled as a multistage stochastic problem [16, 14, 17].

Solving multistage stochastic programs, however, is a challenging numerical problem. The solution of the problem is intractable and common approximations such as stochastic dynamic programming [2] [3] can suffer from high dimensionality (frequently referred as the curse of dimensionality). One method that partially overcomes the curse of dimensionality is the stochastic dual dynamic programming algorithm of [16].

A number of programs implementing SDDP are in-use around the world, ranging from unpublished implementations in academic institutions to professional software such as the product developed by PSR, a software and consulting company, also called *SDDP*[1] [18]. However, until recently, there was no fast, reliable, and open-source implementation of the SDDP algorithm. Without such a tool, researchers and practitioners have not had a common ground for the discussion and analysis of different hydrothermal dispatch formulations and their solutions.

The objective of this work is to build an open source tool, called *HydroPowerModels.jl*, that can be this common ground. *HydroPowerModels.jl* can be used to assess the impact of modeling choices during the planning of a hydrothermal power system. These choices include the usage of different network formulations, the consideration of different risk measures, and the planning horizons for uncertain future costs. Addressing these issues provides the research community and the energy industry with a powerful tool for the efficient design of hydrothermal power systems.

To develop a tool that can be used by both researchers and industry professionals, we take advantage of the Julia language [4] and two main packages: *PowerModels.jl* [7], which implements power flow models for electrical dispatch, *SDDP.jl* [8], which implements the stochastic dual dynamic programming algorithm. Both *PowerModels.jl* and *SDDP.jl* handle their respective optimization models through *JuMP.jl* [9], a Julia package for mathematical optimization. *JuMP.jl* makes it simple to write complex optimization problems and solve them with numerous solvers. *HydroPowerModels.jl* takes advantage from the fact that *PowerModels.jl* and *SDDP.jl* were not only developed in Julia, but also deeply rely on *JuMP.jl* to build and solve mathematical optimization problems. *SDDP.jl* is used to specify the hydro storage dynamics and stochastics of inflows, renewables and loads. *PowerModels.jl* is used to provide multiple network dispatch formulations as a starting point for the *HydroPow-*

[1]Note that the "SDDP" acronym is used to denote a software product, the original algorithm of [16], and a more general class of algorithms inspired by the original SDDP algorithm.

erModels.jl formulation that couples the electrical constraints with hydro constraints and uncertainty.
The next sections are organized as follows: 1) A brief understanding of multistage stochastic problems and decisions under uncertainty; 2) A presentation of the modeling formulation and problem specification; 3) Different simplifications of the model; 4) An explanation of the solution method (SDDP); 5) A comparison of existing alternative solution implementations and listing of the proposed package functionalities; 6) A case study to clarify usage.

2. Multistage Stochastic Optimization

Stochastic Programming (SP) [20] is a branch of optimization under uncertainty, where the realization of some random variables (ω) influence the conditions of the problem and consequently the optimal decisions. Uncertainty relates to the probability distributions of parameters and may be incorporated to the problem in various manners.
One class of simple stochastic programs is known as a two-stage problem with recourse. It can be formulated as follows:

$$1^{\text{st}}\ \text{Stage} \begin{cases} \min\limits_{x} & c^T x + \overbrace{\rho[Q(x,\omega)]}^{\mathcal{Q}(x)} \\ \text{s.t.} & x \in \mathcal{X} \end{cases}$$

$$2^{\text{nd}}\ \text{Stage} \left\{ \mathcal{Q}(x,\omega) = \begin{cases} \min\limits_{u} & g^T u \\ \text{s.t.} & Au = b(\omega) - Ex \\ & u \in \mathcal{U} \end{cases} \right\}.$$

The objective of the first stage is composed of an *immediate cost* term $c^T x$ and a *cost-to-go function* $\mathcal{Q}(x)$. $\mathcal{Q}$ is a function of some decision variables of the first stage x that fix the state of the second stage (its feasible region). These variables are called *state variables*. First stage decisions are made under uncertainty, while the decision variables of the second stage u are chosen after the realization of the variable ω (under a deterministic scope) and are so called *recourse variables*. The function ρ is known as a *risk measure* [1] and is commonly assumed to be the expectation operator $\mathbb{E}$.
The goal of this problem is to find an optimal stage decision x and an optimal second-stage decision u for each realization of ω conditioned on x. Collectively, this set of decisions is known as a *policy*.
The two-stage problem discussed above naturally extends to a *multistage* problem via recursion. A multistage stochastic program with T stages can be formulated as follows:

$$\begin{aligned} &\min_{x_1 \in \mathcal{X}_1} f_1(x_1) + \\ &\qquad \rho[\inf_{x_2 \in \mathcal{X}_2(x_1,\omega_2)} f_2(x_2,\omega_2) + \\ &\qquad\qquad \rho[\ldots + \rho[\inf_{x_T \in \mathcal{X}_T(x_{T-1},\omega_T)} f_T(x_T,\omega_T)]\ldots]] \end{aligned}$$

Assuming the problem is linear, we have:

$$\begin{aligned} &\mathcal{X}_t(x_{t-1},\omega) = \left\{x_t \geq 0 : A_t x_t = b_t(\omega) - E_t x_{t-1}\right\} \\ &f_t(x_t,\omega) = c_t^T x_t \end{aligned}$$

In this setting, the uncertain data $\omega_1, \ldots, \omega_T$ is revealed gradually over time. The sequence $\omega_t \in \mathbb{R}^{d_t}$ of data vectors is viewed as a stochastic process, i.e., as a sequence of random variables with a specified probability distribution.
Just like in the two stage problem, there are state variables at every stage that partially impact the objective through the risk measure of the subsequent stages optimal values. The policy optimized by this problem is a mapping from the realized stochastic process to the decisions for each stage.
Using dynamic programming [2], the nested formulation of a multistage stochastic program may be represented by the following Bellman recursion for each stage:

$$Q_t(x_{t-1},\omega_t) = \left\{ \begin{cases} \min\limits_{x_t} & c_t^T x_t + \rho_{t+1}[Q_{t+1}(x_t,\omega_{t+1})] \\ \text{s.t.} & A_t x_t = b_t(\omega_t) - E_t x_{t-1} \qquad [\pi_t(\omega_t)] \\ & x_t \geq 0 \end{cases} \right\}$$

For the purpose of simplicity, we assume that $\mathcal{Q}_{T+1}(\cdot,\cdot) = 0$.
In these equations, the optimal value at stage t depends on previous decision x_{t-1} and the realization of the data process ω_t. Finally, the optimal value of the first stage problem gives the optimal value of the corresponding multistage problem.

3. The Hydrothermal Dispatch Problem

The hydrothermal dispatch problem is a multistage stochastic optimization problem that comprises an optimal power flow (OPF) problem and the hydro storage management for multiple periods and scenarios.
Introduced by [6], the OPF problem extends the economic dispatch problem, where the goal is to plan the operation of an electrical power system by determining the contribution of available energy sources in supplying demands, to include constraints representing the power flow equations, resulting in a more realistic model of the dispatch.
Since its introduction, the OPF problem has received additional constraints to better represent power systems. The adopted version of the *PowerModels.jl* package, is a AC Optimal Power Flow (AC-OPF) problem. The AC Power Flow constraints implement voltage bounds, generation bounds, nodal conservation of power, power flow on lines and thermal limit of the lines (power flow limit) for an alternating current system. Different from original OPF formulations, support was also added for multiple load and shunt components on each bus together with a line charging that supports conductance and asymmetrical values.
Besides the modifications from *PowerModels.jl*, deficit variables to work as slack to the attendance of each load have frequently been used by the electrical sector and will be incorporated in our package. This variables allow a more detailed evaluation of the cost of not attending some demands.
The optimal dispatch tries to find the most economic use of system resources and is the objective function of an OPF. However, in a hydrothermal power system, where water is a main resource and its inflow is uncertain, risk averse planning is an important task to ensure the lowest cost operation. The necessary hydro storage management to attend demand during periods of scarcity adds an extra layer of decisions and constraints.
Reservoir operation is a problem composed by the balance of incoming and outgoing flow of water as well as its usage to generate power. Since many storage facilities are linked through rivers, this problem may be simply viewed as a directed flow graph where each node in the graph represents a storage reservoir. A constraint binds the power generation of generators using water as fuel to the out-

flow of the respective storage. Operation is also restricted by the limits of reservoir volume and outflow.
The solution of this problem returns an operation policy representing the optimal generation and hydro management possible given the horizon studied and the scenarios considered.
As it is common for multistage problems, we will define the underlying sub-problem (i.e., $\mathcal{Q}$) that unifies the OPF problem with the hydro-thermal dispatch problem.

3.1 The Mathematical Model

Following the notation chosen by the *PowerModels.jl* package, the sets and parameters used to define the sub-problem are listed with the addition of those created for the hydro storage management and the data for the state of the sub-problem:

Sets.

N - buses
R - reference buses
E, E^R - branches, forward and reverse orientation
G, G_i - generators and generators at bus $i \in N$
L, L_i - loads and loads at bus $i \in N$
S, S_i - shunts and shunts at bus $i \in N$
H, H^G - reservoirs and reservoirs with power generation
$H_h^U, H_h^{\mathfrak{S}}$ - upstream reservoirs out-flowing and spilling to $h \in H$
G_h^H - generator at reservoir $h \in H$

Data.

S_k^{gl}, S_k^{gu} $\forall k \in G$ - generator complex power bounds
c_{2k}, c_{1k}, c_{0k} $\forall k \in G$ - generator cost components
v_i^l, v_i^u $\forall i \in N$ - voltage bounds
S_k^d $\forall k \in L$ - load complex power consumption
Y_k^s $\forall k \in S$ - bus shunt admittance
Y_{ij}, Y_{ij}^c $\forall (i,j) \in E$ - branch pi-section parameters
T_{ij} $\forall (i,j) \in E$ - branch complex transformation ratio
s_{ij}^u $\forall (i,j) \in E$ - branch apparent power limit
i_{ij}^u $\forall (i,j) \in E$ - branch current limit
$\theta_{ij}^{\Delta l}, \theta_{ij}^{\Delta u}$ $\forall (i,j) \in E$ - branch voltage angle difference bounds
$\overline{\nu}_h$ $\forall h \in H$ - reservoir volume limit
$\overline{u}_h$ $\forall h \in H$ - reservoir outflow limit
p_h $\forall h \in H^G$ - hydro generation production factor
c_d - deficit cost
c_s - spillage cost

Operators.

$\Re$ - real part of a complex number
$\angle$ - angle of the polar representation of a complex number
$(\cdot)^*$ - complex conjugate
$|\cdot|$ - absolute value

States.

$\nu_{h,t-1}$ $\forall h \in H$ - incoming reservoir volume
a_h $\forall h \in H$ - reservoir inflow

For simplicity the stage index t of the sub-problem is omitted from every data and variable with exception of the incoming reservoir volume $\nu_{h,t-1}$ (in order to differentiate it from the outgoing reservoir volume ν_h).
Notice that, as usual, alternating components (i.e. voltage, power generation, power demand and power flow) are modeled using complex numbers to fully represent the process.
Bellow, the decision variables of our problem:

Variables.

S_k^g $\forall k \in G$ - generator complex power dispatch
V_i $\forall i \in N$ - bus complex voltage
S_{ij} $\forall (i,j) \in E \cup E^R$ - branch complex power flow
U_h $\forall h \in H$ - reservoir outflow
$\mathfrak{S}_h$ $\forall h \in H$ - reservoir spillage
ν_h $\forall h \in H$ - reservoir volume
D_i $\forall i \in N$ - bus complex deficit

A complete mathematical model of the sub-problem is as follows,

$$\mathcal{Q}_t(\nu_{t-1}, \omega_t) =$$

$$\text{minimize: } \sum_{k\in G} c_{2k}(\Re(S_k^g))^2 + c_{1k}\Re(S_k^g) + c_{0k} + \sum_{i\in N} c_d \Re(D_i) + \sum_{h\in H} c_s \mathfrak{S}_h + \rho_{t+1}[\mathcal{Q}_{t+1}(\nu_t, \omega_{t+1})] \tag{1a}$$

subject to:

$$\angle V_r = 0 \ \ \forall r \in R \tag{1b}$$

$$D_i + \sum_{k\in G_i} S_k^g - \sum_{k\in L_i} S_k^d - \sum_{k\in S_i} Y_k^s |V_i|^2 = \sum_{(i,j)\in E_i \cup E_i^R} S_{ij} \ \ \forall i \in N \tag{1c}$$

$$S_{ij} = \left(Y_{ij} + Y_{ij}^c\right)^* \frac{|V_i|^2}{|T_{ij}|^2} - Y_{ij}^* \frac{V_i V_j^*}{T_{ij}} \ \ \forall (i,j) \in E \tag{1d}$$

$$S_{ji} = \left(Y_{ij} + Y_{ji}^c\right)^* |V_j|^2 - Y_{ij}^* \frac{V_i^* V_j}{T_{ij}^*} \ \ \forall (i,j) \in E \tag{1e}$$

$$|S_{ij}| \le s_{ij}^u \ \ \forall (i,j) \in E \cup E^R \tag{1f}$$

$$\theta_{ij}^{\Delta l} \le \angle(V_i V_j^*) \le \theta_{ij}^{\Delta u} \ \ \forall (i,j) \in E \tag{1g}$$

$$\nu_h + U_h + \mathfrak{S}_h = \nu_{h,t-1} + a_h(\omega_t) + \sum_{k\in H_h^U} U_k + \sum_{k\in H_h^{\mathfrak{S}}} \mathfrak{S}_k \ \ \forall h \in H \tag{1h}$$

$$U_h p_h = \Re(S_{G_h^H}^g) \ \ \forall h \in H^G \tag{1i}$$

$$S_k^{gl} \le S_k^g \le S_k^{gu} \ \ \forall k \in G \tag{1j}$$

$$v_i^l \le |V_i| \le v_i^u \ \ \forall i \in N \tag{1k}$$

$$0 \le \nu_h \le \overline{\nu}_h \ \ \forall h \in H \tag{1l}$$

$$0 \le U_h \le \overline{u}_h \ \ \forall h \in H \tag{1m}$$

[2]

The objective of the sub-problem (1a) is to minimize the costs of real power generation, cost of deficit from the real energy supply, cost of spillage (in order to try and avoid degenerate solutions) and the cost-to-go function $\mathcal{Q}$.

Constraint (1b) fixes reference buses complex voltage angles to zero, as the remaining angles will be defined accordingly.

Constraint (1j) bounds the complex power generation, representing the physical limitation of generators and fuel source availability.

The magnitude of the complex voltage is bounded in constraint (1k) by restricting the absolute square of its value. The upper limit alone defines a circular feasible region for each voltage, while the lower limit reshapes the region as a ring, bringing a non-convexity to the problem.

The Branch complex power flow is formulated in (1d) and (1e), dependent on the voltage at each end and implementing elements of line charging and the effects of transformers. The power flow is bounded in (1f) through its absolute value. These power limits of the lines represent thermal limits and stability limits.

[2]Note that the elements of E are pairs of elements of N, so, in equations (1d) and (1e), variables defined over N are indexed by elements of E.

Constraint (1c) implements Kirchhoff's Current law (KCL), which refers to power preservation at each node, balancing generation, demand, flow and shunt. Although, deficit variables have been added in order to guarantee feasibility in case of lack of power availability.

Angle difference between buses are bounded in (1g). The reason for the limits is to approximate the transient stability constraints of power flowing in branches. These restrictions refer to the synchronism among machines at each end of a line. The limits depend on the equipment installed and the system configuration.

An important variable in an economic dispatch problem is the marginal cost of energy at each bus, which, in optimality, is determined by the dual value of (1c). This value is also referred to as a shadow price, local marginal price (LMP), or nodal price. Regardless, this value represents the cost of an extra unit of energy in a bus.

The conservation of water equation is implemented in (1h), where the water stored at a reservoir should equal the water previously stored plus the incoming flow (precipitation and water from upstream reservoirs), minus the portion used to generate energy and the one spilled away.

The binding of the hydro real power generation and water used is done in (1i), which depends on the efficiency of the generator modeled through a production factor.

Constraints (1l) and (1m) bound respectively the volume of water stored and the amount of water used in generation. These limits are defined by the capacity limit of storage facility and the equipment installed.

Note that problem (1) is a nonlinear, non-convex optimization problem. This will have important implications for our solution approach.

4. Network Formulations

The AC Optimal Power Flow problem, defined in (1), captures the nonlinear and complex nature of the power flow. However, solving a nonlinear problem (NLP) is hard. Guaranteeing global optimality often involves an extensive number of comparisons and, thus, it is not done. Local optimums are used instead, providing inconsistent solutions. Along with the scarcity of efficient NLP solvers and the numerical issues created by large problem instances, the depth of research using AC-OPF is reduced. Besides, the requirement of convexity frequently needed for applications limit its usage. Hence, many approximations and relaxations have been developed for the AC-OPF [15]. In general these are simplifications and are easier to solve, but they ignore some parts of the problem. As a result, it is imperative to understand and weigh the advantages and compromises of each formulation when choosing one to use in a model.

The *Linear DC approximation* is a linear formulation which partially represents power voltage. This formulation makes some assumptions for linearization purposes: voltage magnitudes are sufficiently near nominal value (one), angle differences are close to zero, and there are practically no power losses. In all effects, a purely active power system linear model. Still it is important to notice that, while this model seeks to approximate the feasible region of the AC power flow, it may not include the entire feasible region, including the global optimum.

Instead of approximating parts of the power system, it is possible to relax some of the nonlinear power flow constraints. The resulting *relaxations*, when solved to optimality, provide lower bounds to the original problem because their feasible sets include all the solutions of the original problem. Different convex relaxations were proposed for the optimal power flow problem, here we highlight

some of the most important ones. Convex relaxations are specially useful because their solution can be proved to be optimal.
The *Copper Plate* is the simplest linear relaxation and models the system by a centralized energy pool, relaxing transmission lines limits and the Kirchhoff's laws. Easily implementable and solvable, this is the most simplified linear formulation, neglecting the entire grid of the problem. However, because of the simplifications of this model it produces only one shadow price, i.e. the marginal price of energy in each node of the network are equal. This may send a distorted signal to the agents responsible for the system, and may return the most unfeasible solution across all formulations.
Another linear relaxation, the *Transportation model*, or tube model, extends Copper Plate by adding line limits. Locational restrictions are better represented and the value of transmission lines are clearer. Although, by completely ignoring the power voltage, network design becomes less relevant and incorrect system analysis is possible.
The *SOC relaxation* is a non-linear convex relaxation that is tighter than the linear versions, i.e., its feasible region is strictly contained inside them. This formulation relaxes the non-convex constraints of the problem, composed by the bilinear product of the voltage variables, by neglecting the phases of the voltages and saving only their branch wise difference and magnitudes. The resulting problem may be specified as a second order cone formulation, hence its name.
The *SDP relaxation* deals with the non-convex constraints of the problem by using the fact that they define a positive semi-definite matrix with rank 1. The relaxation comes from removing the rank 1 restriction. The feasible region of this formulation is contained within the feasible region of the SOC relaxation, providing a better bound to the original problem.
The *QC relaxation* exploits the polar form of the non-convex constraints and uses known convex envelopes to relax each non-convex term present. These envelopes retain stricter links between the voltage variables, producing a tighter relaxation than the SOC formulation.
These, and other relaxations, provide a good alternative to solve the original problem. Besides bounding the optimal value of the original problem, they have sufficiently good solutions for real world applications. In fact, by being convex, these relaxations are suitable to be used in many solution methods for multistage problems, for instance SDDP for which convergence is guaranteed if subproblem are convex.
The *PowerModels.jl* package, a framework for steady-state power network optimization, is able to construct these and other mathematical programming formulations of the OPF problem. Since *PowerModels.jl* uses *JuMP.jl* to construct these formulations, they can be passed to a variety of solvers. This allows the user to easily choose an approximation or relaxation, solve it, and then discuss and compare the impacts of using different relaxations and approximations in the planning and operation of the economic dispatch problem.

5. Solution method

As we have seen, the hydrothermal dispatch is a complicated problem with different network formulations. The OPF problem, is only a part of a sub-problem composing a multistage stochastic program. As discussed previously, solving a multistage stochastic program has its own difficulties, and requires specific and efficient algorithms.
In a multistage stochastic program, we are faced with a cost-to-go function: $\rho_{t+1}[Q_{t+1}(x_t, \omega_{t+1})]$. The issue is that $\rho_{t+1}[Q_{t+1}(x_t, \omega_{t+1})]$ also depends on a cost-to-go $\rho_{t+2}[Q_{t+2}(x_{t+1}, \omega_{t+2})]$, and the evaluation of those functions can be expensive.
The crucial step that facilitates the solution of these problems is to construct approximations of the cost-to-go functions, recursively, going backward in time. Thus, the optimal value of the first stage problem approximates the optimal value of the corresponding multistage problem.
For the construction of this approximation a widely used method is dynamic programming, which evaluates the function in a range of discrete values of the state variable for further interpolation. However, this method becomes intractable with the growth of the state dimension (commonly referred as the *curse of dimensionality* of dynamic programming). A solution for this was proposed by [16] with a method called stochastic dual dynamic programming (SDDP).
The methodology, simply posed, approximates the cost-to-go function by the maximum of a set of linear hyper-planes called *cuts*.
SDDP is based on an interactive construction of the cost-to-go function approximations. The procedure may be divided in subsequential forward and backward passes, where the first chooses points in which the second will update the current approximation of the functions. For a detailed discussion of the SDDP algorithm, see [16, 8].
Other solution algorithms have been proposed to solve multistage problems, such as progressive hedging and nested Benders. However, for most large instances, none has proven to be viable and efficient alternatives. SDDP has been extensively used to solve and plan hydrothermal dispatch operations since its original publication.
On the other hand, this method is a complex and difficult to implement algorithm. Moreover, to make it flexible enough for different applications while not compromising performance is a main issue. Hence, the historical scarcity of reliable and open software versions has limited development and discussions of its contributions. Yet, new and efficient implementations have risen with the advance of open-source languages as the Julia language.
One such implementation is *SDDP.jl*, a Julia/JuMP package for solving large multistage convex stochastic optimization problems using SDDP. It provides a practical and efficient way to find the solution to the hydrothermal dispatch problem. With this package it was possible to define the dynamics and random variables of our problem and solve the instantiated model with the SDDP algorithm.

6. HydroPowerModels.jl

HydroPowerModels.jl uses the *PowerModels.jl* and *SDDP.jl* packages to implement and solve different hydrothermal dispatch formulations. It provides an interface to easily solve and simulate hydrothermal dispatch models and allows the creation of a collection of hydrothermal problems described in input files for the package (following the *PowerModels.jl* standard), thereby helping the discussion of methodology and the resulting policies for specific case instances.
In contrast to the previously available official software for hydrothermal dispatch models, the proposed package is part of an academic open-source effort. This helps to promote the continuous improvement of models and solution algorithms for the research community.
Other academic implementations of SDDP have being developed and may be applied to the hydrothermal dispatch problem. However, the advances of the Julia Language and *JuMP.jl* are a much more adequate framework than those of MATLAB [5] or Python [19, 12]. Moreover, a free and open-source tool can be of great help

for the research community alternatively to commercial solvers as [13] and the renowned version from *PSR Inc*.
Additional implementations of SDDP are also available in Julia [10] [11], but *SDDP.jl* [8] has proven an easy to learn, efficient version of SDDP that is flexible enough for the purposes of the *HydroPowerModels.jl* package.
HydroPowerModels.jl is composed of different and useful functionalities, from compact case sharing to dispatch solution results visualizations. A work-flow of a simple usage of the package helps to give a basic overview:

—Load case data from input files describing: Power network data; Reservoir facilities details and water network data; Inflow scenarios.

—Receive case parameters: Power network formulation; Number of stages; Number of hours in Stage and optimizer to solve the sub-problems.

—Build the multistage, hydrothermal steady-state power network optimization problem.

—Run SDDP method to approximate the cost-to-go functions.

—Simulate the policy.

A code example is presented in the next subsections to help clarify the usage of the package. Although, for a more extensive tutorial of the package, a detailed documentation is made available in `https://andrewrosemberg.github.io/HydroPowerModels.jl/latest/`.

6.1 Usage

The usage of *HydroPowerModels.jl* follow the paradigms of the Julia language and the structure of the dependent packages.
In order to access the available functionalities, first import *HydroPowerModels.jl* and an adequate solver:

```
using HydroPowerModels
using GLPK
```

Load a case by passing the folder containing the input files (`PowerModels.json`, `hydro.json`, and `inflows.csv`):

```
data = HydroPowerModels.parse_folder(
    "case3_folderpath")
```

Use `create_param` to create a set of problem parameters. For example, a 12-stage problem using the DC approximation can be specified as follows:

```
params = create_param(
 stages       = 12,
 stage_hours  = 168.0,
 model_constructor_grid  = DCPPowerModel,
 optimizer    = with_optimizer(GLPK.Optimizer))
```

Then, build the Model and execute the SDDP `train` method:

```
m = hydrothermaloperation(data, params);

HydroPowerModels.train(m)
```

Finally, simulate the performance of the policy with 1000 Monte Carlo scenarios:

```
# Simulate 1000 instances
results = HydroPowerModels.simulate(m, 1000);
```

```
 Dict{Any,Any} with 5 entries:
"simulations" => Dict[Dict{Any,Any}(Pair{Any,...
"data"        => Dict{Any,Any}[Dict{Any,Any}(...
"params"      => Dict{Any,Any}(Pair{Any,Any}(...
"machine"     => Dict("cpu"=>"Intel(R) Xeon(R)...
"solve_time"  => 205.31247
```

7. Case Study

For a case study, consider a the hydrothermal dispatch of a realistic system with the following specifications:

—Number of buses: 166

—Number of loads: 286

—Number of generators: 145

—Number of branches: 235

In order for a qualitative view of the system, the package disposes a graph illustration plot:

```
plot_grid(data, node_label=false)
```

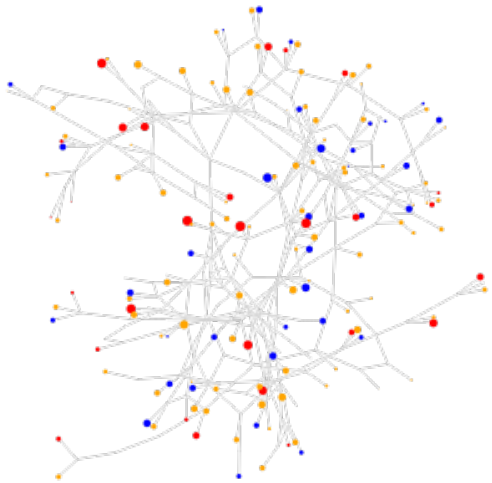

Fig. 1. Network Grid graph

Figure 1 shows the installed power available in the network (grouped by bus) using a logarithmic scale. The red nodes represent the thermal generators, the blue represent the hydro genera-

tors. For comparison purposes, orange nodes have been added that are equivalent to average real power demands.
As we can see from the plot of the grid 1, this appears to be a well balanced case, with similar installed hydro and thermal power capacity and with a reasonable average demand. In addition, it is a well distributed network, without any evident critical sections susceptible to impacting power flow problems. Those facts are indications of a significant hydro-generation optimal dispatch without large complications.
For this study, a 52 stage horizon planning and simulation have been executed using the following case parameters:

—Number of stages: 52

—Number of hours in stage: 168

—Network Formulation: Transportation model relaxation

7.1 Results

The simulate command returns a detailed dictionary of the execution. In order to plot those results returned by the simulate function, you may choose from a variety of methods, including the function `plot_aggregate_results()`. This function receives the dictionary results and generates the most common aggregated variable plots, which best summarize simulations of a hydrothermal dispatch:

```
plot_aggregated_results(results)
```

Figures 2 to 9 show the output from the above command. As mentioned, the plots are of aggregated quantities, but the methods used to aggregate were chosen in order to help analysis. For example: the final nodal price is an average of nodal prices weighted by the contribution of local loads to the total demand; reservoir volume was grouped weighted by the amount of energy that could be produced by the stored water (as was the inflow of water).
As expected the optimal dispatch of the simulations uses more hydro-generators, however it needs thermal-generators to met all demands without deficit. On this hydro-dominated system, the uncertain inflow is a driving factor of optimal dispatch. As we can see in Figure 9, the inflow has a strong seasonality component, resulting the significant seasonality trait observable in the variables of the policy simulations. 2-3, 5-8.
Similar studies are possible for any case and formulation chosen, helping to analyze existing realistic cases and assess impacts of future system changes. For example, we reran the study using a SOC relaxation to compare results. Most quantities analyzed, as those presented in the aggregated results, are similar, but operation costs differ as it is observable in figures 10,11. This is expected, since the SOC relaxation is a more realistic representation of reality.
Moreover, it is important to point out this extra cost would be even bigger if we used the policy of the transportation model to operate the SOC model in the simulations. In other words, there is a hidden cost when planning a dispatch associated with the fact we are not evaluating costs on the actual system we want to operate.
Therefore, measuring the impacts of possible simplifications is a needed step in discussing hydrothermal economic dispatch. *HydroPowerModels.jl* intends to provide a common ground for discussions and analysis and a easy to use tool for research and applications.

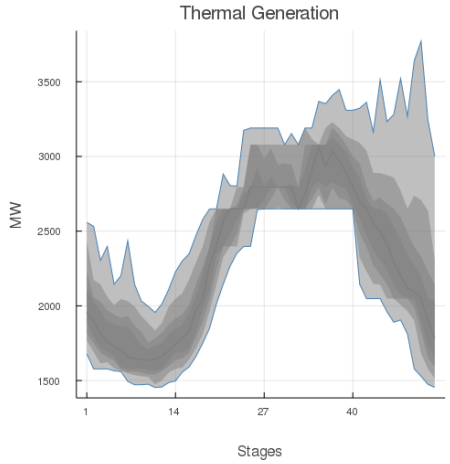

Fig. 2. Thermal Generation

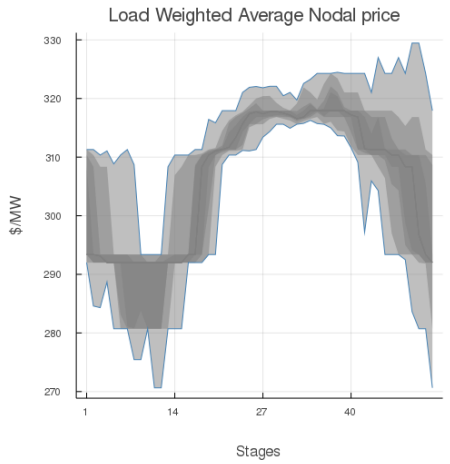

Fig. 3. Load Weighted Average Nodal Price

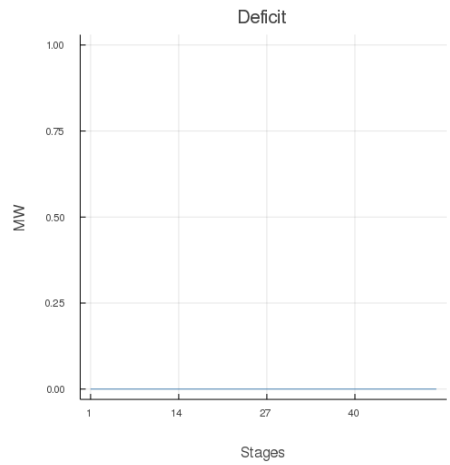

Fig. 4. Deficit

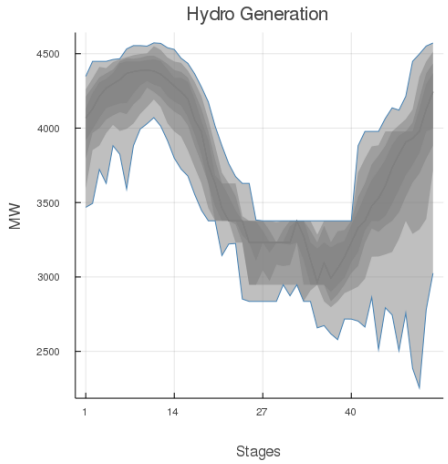

Fig. 5. Hydro Generation

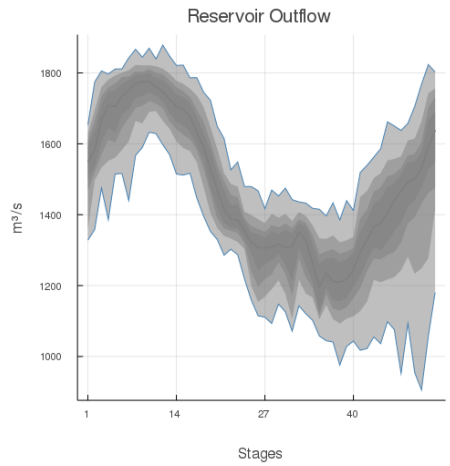

Fig. 6. Reservoir Outflow

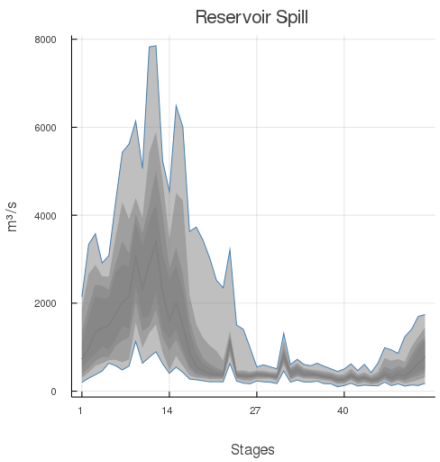

Fig. 7. Reservoir Spillage

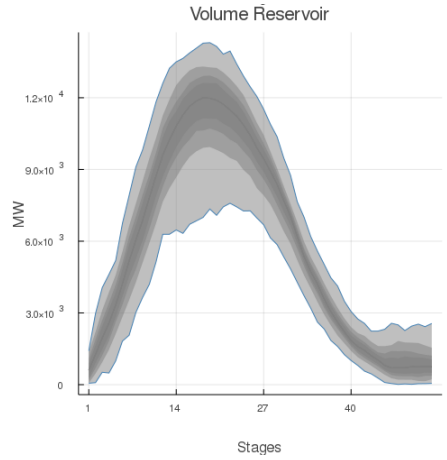

Fig. 8. Volume Reservoir

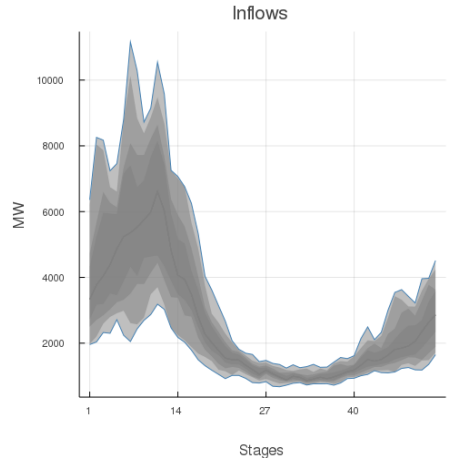

Fig. 9. Inflow

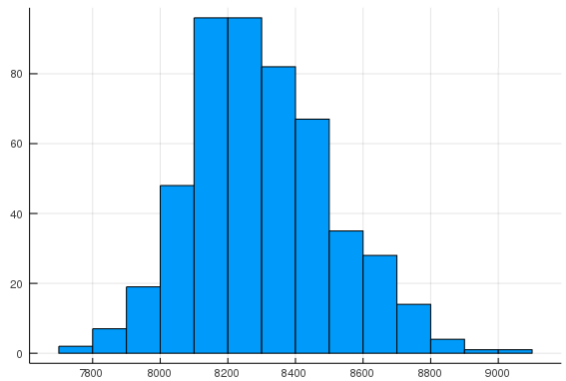

Fig. 10. Simulation Costs Transportation

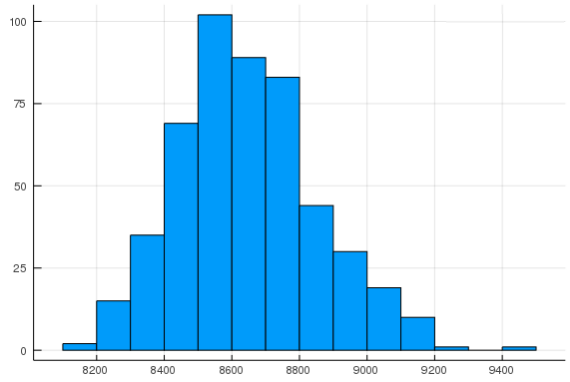

Fig. 11. Simulation Costs SOC

8. References

[1] Philippe Artzner, Freddy Delbaen, Jean-Marc Eber, and David Heath. Coherent measures of risk. *Mathematical finance*, 9(3):203–228, 1999.

[2] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[3] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[4] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[5] Léopold Cambier and Damien Scieur. leopoldcambier/fast: v0.9.1b, March 2018.

[6] J Carpentier. Contribution to the economic dispatch problem. *Bulletin de la Societe Francoise des Electriciens*, 3(8):431–447, 1962.

[7] Carleton Coffrin, Russell Bent, Kaarthik Sundar, Yeesian Ng, and Miles Lubin. Powermodels.jl: An open-source framework for exploring power flow formulations. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–8, June 2018.

[8] Oscar Dowson and Lea Kapelevich. SDDP.jl: a Julia package for Stochastic Dual Dynamic Programming. *Optimization Online*, 2017.

[9] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[10] Vincent Leclère, Henri Gérard, François Pacaud, and Tristan Rigaut. Stochdynamicprogramming. jl a julia library for multistage stochastic optimization.

[11] Benoît Legat. Juliastochopt/structdualdynprog.jl: v0.2.0, October 2018.

[12] Alexander Shapiro Lingquan Ding, Shabbir Ahmed. A python package for multi-stage stochastic programming, May 2019.

[13] N Löhndorf. Quasar optimization software 2.4, 2018.

[14] MEP Maceiral, DDJ Penna, AL Diniz, RJ Pinto, ACG Melo, CV Vasconcellos, and CB Cruz. Twenty years of application of stochastic dual dynamic programming in official and agent studies in brazil-main features and improvements on the newave model. In *2018 Power Systems Computation Conference (PSCC)*, pages 1–7. IEEE, 2018.

[15] Daniel K Molzahn, Ian A Hiskens, et al. A survey of relaxations and approximations of the power flow equations. *Foundations and Trends® in Electric Energy Systems*, 4(1-2):1–221, 2019.

[16] Mario VF Pereira and Leontina MVG Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical programming*, 52(1-3):359–375, 1991.

[17] Andy Philpott. On the Marginal Value of Water for Hydroelectricity. In Tamás Terlaky, Miguel F. Anjos, and Shabbir Ahmed, editors, *Advances and Trends in Optimization with Engineering Applications*, pages 405–425. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017.

[18] PSR. Software | PSR, 2019. [Online; accessed 2019-07-06].

[19] Luciano Raso, David Dorchies, Jan Kwakkel, and Pierre Olivier Malaterre. Optimist: a python library for water system optimal operation and analysis using sddp. 2016.

[20] Alexander Shapiro, Darinka Dentcheva, and Andrzej Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.