HydroPowerModels.jl

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

[1]Pontifical Catholic University of Rio de Janeiro
[2]PSR Consulting, Inc
[3]Northwestern University, Evanston, IL
†Work supported by CAPES Foundation
†Work supported by FGV Energia's project: P&D PD-09344-1703/2017

October 11, 2019

Agenda I

1. Introduction
   - Author
   - Overview
2. Hydrothermal Economic Dispatch
3. Dependencies and Integration
   - PowerModels.jl
   - SDDP.jl
   - Hydro
   - Integration
4. Example
   - Real-Life case study
   - HydroPowerModels.jl Usage
5. Bibliography

Introduction: Andrew Rosemberg

- Control Engineering at Pontifical Catholic University of Rio de Janeiro (PUC-RIO), Brazil.
- Double Degree General Engineering at École centrale de Marseille, France.
- Currently enrolled in the Operations Research Masters at PUC-RIO (Electrical Department).
- Researcher at Laboratory of Applied Mathematical Programming and Statistics (LAMPS).

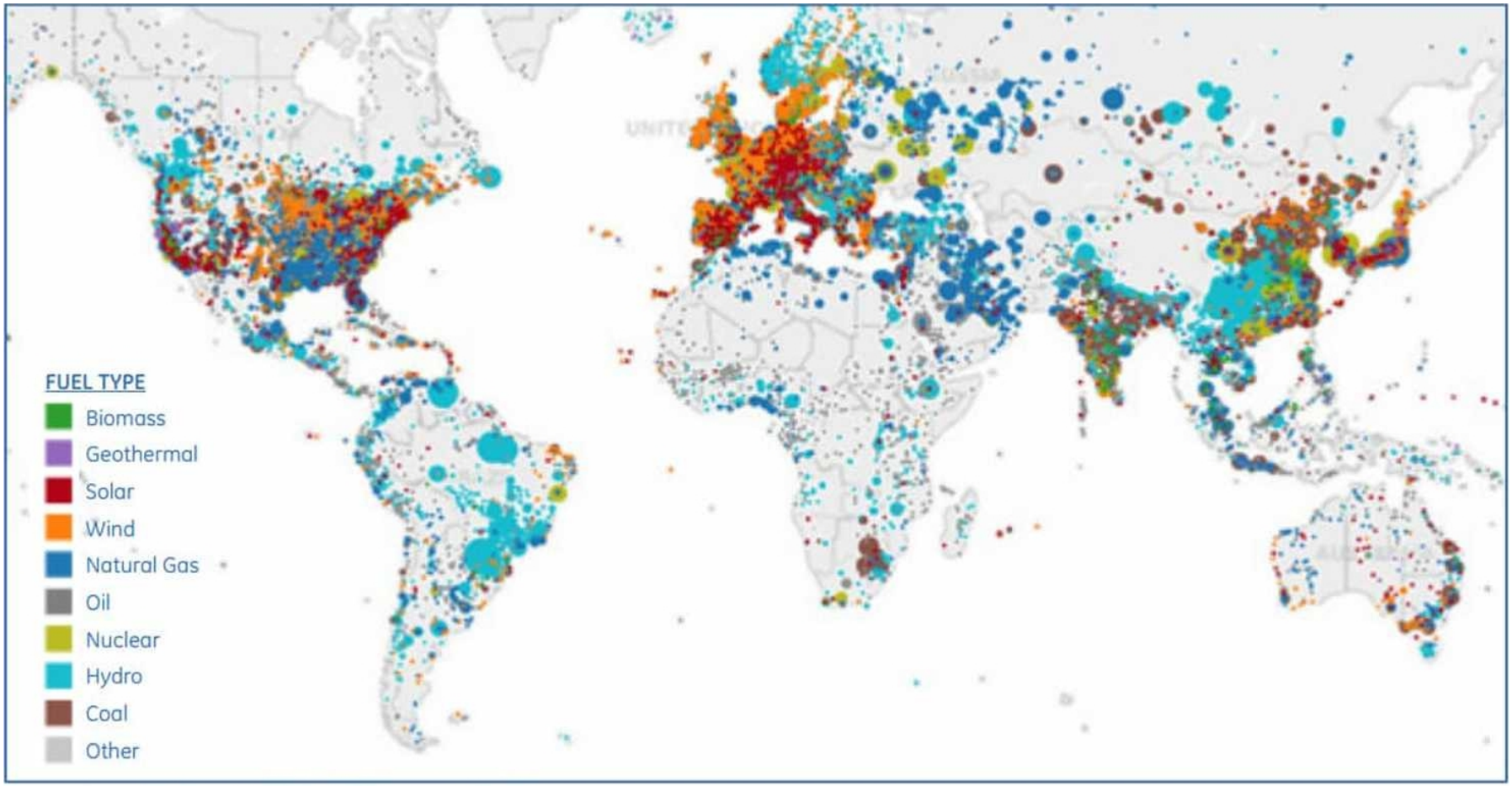
Overview

- **HydroPowerModels.jl** is a Julia Programming package for solving multistage, steady-state, hydro-dominated, power network optimization problems with Stochastic Dual Dynamic Programming (SDDP) [Pereira and Pinto, 1991].
- Mathematical programming modeling is done through **JuMP.jl** [Dunning et al., 2017].
- Optimal Power Flow (OPF) Problem Specifications and Network Formulations are handled by the **PowerModels.jl** package [Coffrin et al., 2018].
- Solution method is handled by the **SDDP.jl** package [Dowson and Kapelevich, 2017].

Hydrothermal Economic Dispatch

- Important for the planning and operation of hydro-dominated electrical systems such as the Brazilian national grid.
- Objective: Coordinate generation, energy distribution, and hydro-storage management in order to minimize the cost of operation.
- Uncertainties: Hydrology (i.e., rainfall and other inflows) of the hydroelectric plants.
- Often insufficient capacity of thermal generation to meet demand.
- Trade-off between using water for cheap generation in the present, against conserving water for future periods of drought.
- The hydrothermal dispatch problem is often modeled as a multistage stochastic problem Pereira and Pinto [1991], Maceiral et al. [2018], Philpott [2017].

Is it Relevant?



Source: Power plant data source Platts UDI Database, June 2012
Note: Circle size represents installed capacity (MW).

Optimal Power Flow

- Plan the operation of an electrical power system.
- Power Flow Equations.
- AC OPF.
- Approximations and Relaxations.

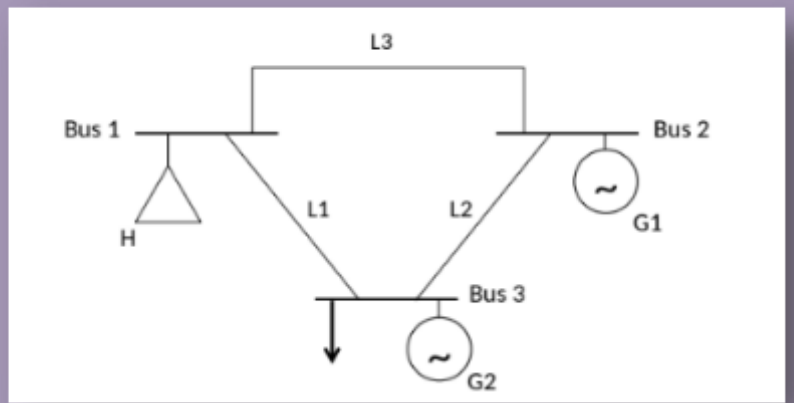
PowerModels.jl

- Steady-State Power Network Optimization Framework.
- Parsing and modifying network data .
- Enable computational evaluation of emerging power network formulations and algorithms.
- Models different Network Formulations (e.g. AC, DC-approximation, SOC-relaxation, ...).

PowerModels.jl

Network Description

Case.m



AC, DC, SOC, ...

PF,OPf, ...

Network Optimization Model

JuMP

$$\min_{x \in X} \quad f(x)$$
$$s.t. \quad g(x) \in \aleph$$

Solution Method

- Solving multistage stochastic programs is a challenging numerical problem.
- The solution of the problem is intractable.
- Common approximations (like SDP) - Curse of dimensionality.
- Overcame by SDDP.

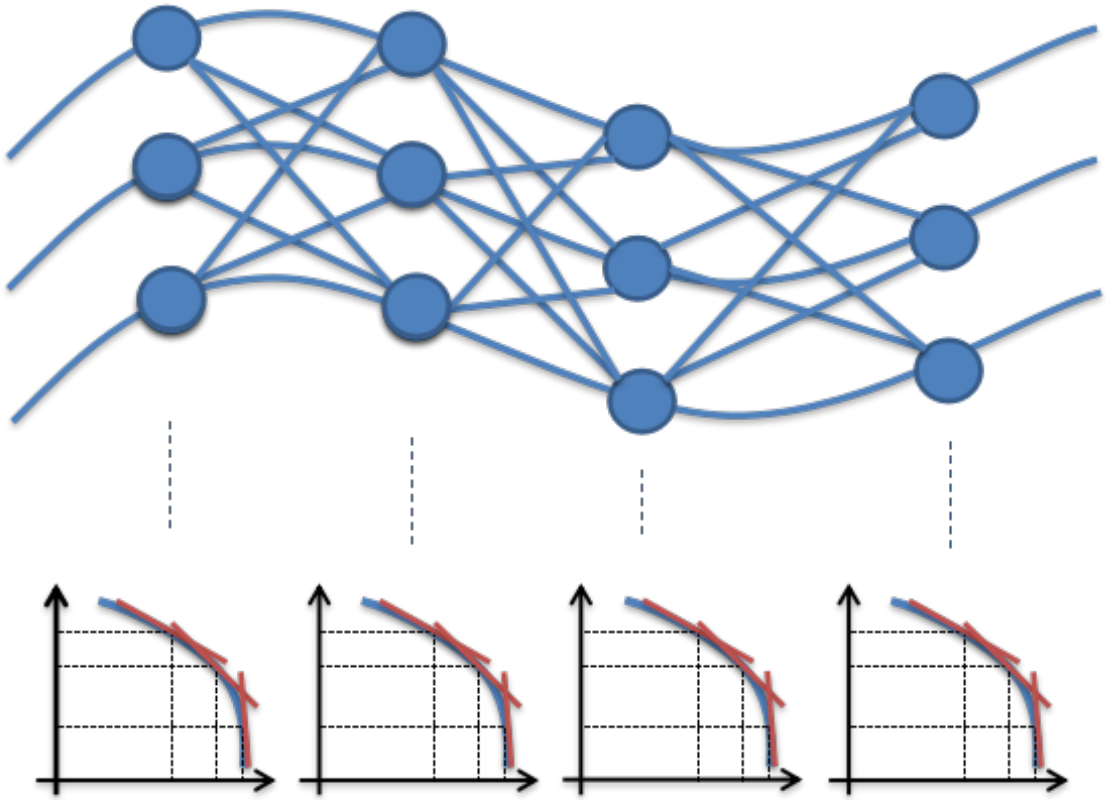
SDDP.jl

- Julia/JuMP Package for solving large multistage convex stochastic optimization problems using Stochastic Dual Dynamic Programming.
- Open source.
- Generic (Not domain specific).

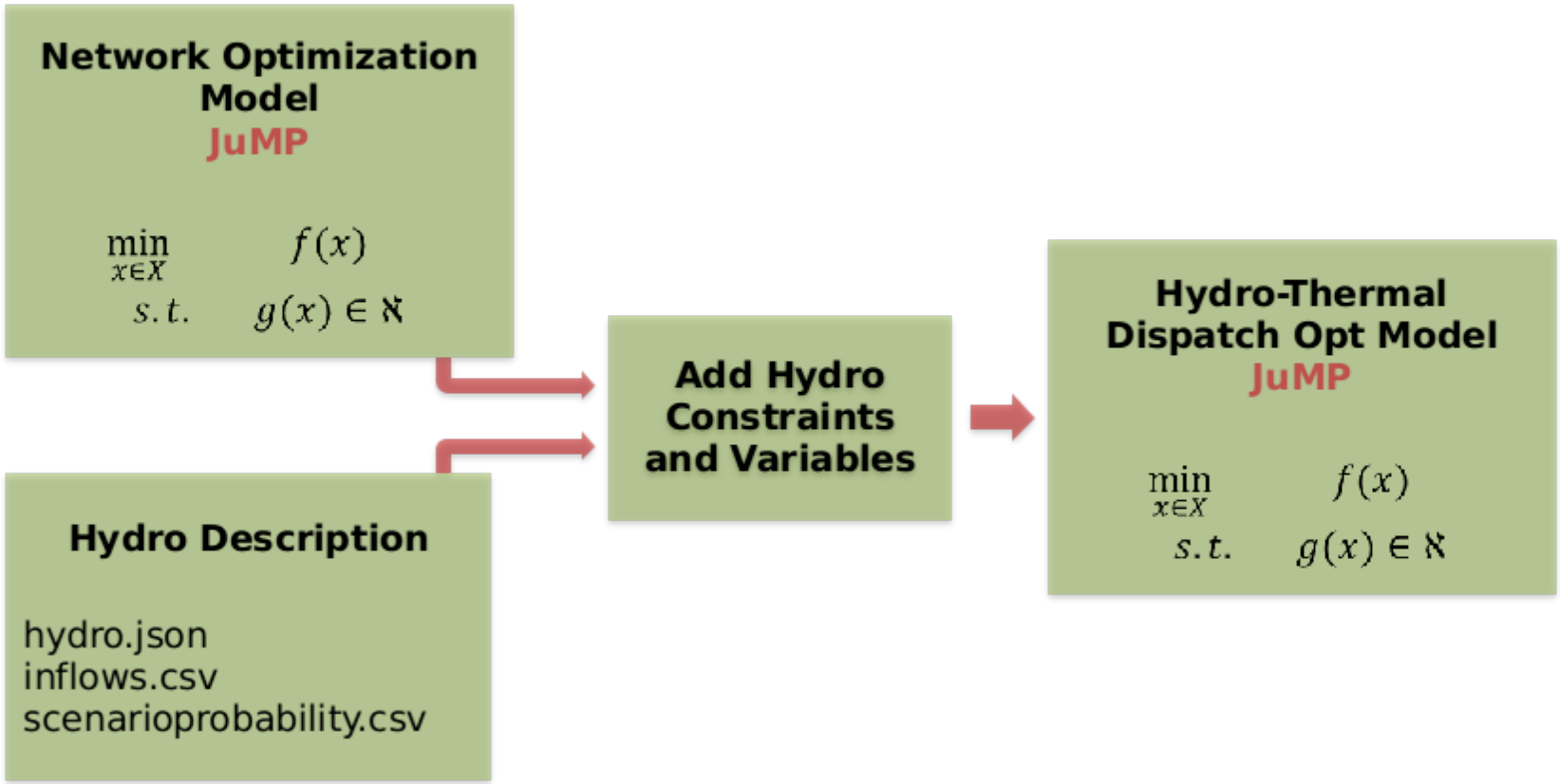
SDDP.jl

Optimization Model
JuMP

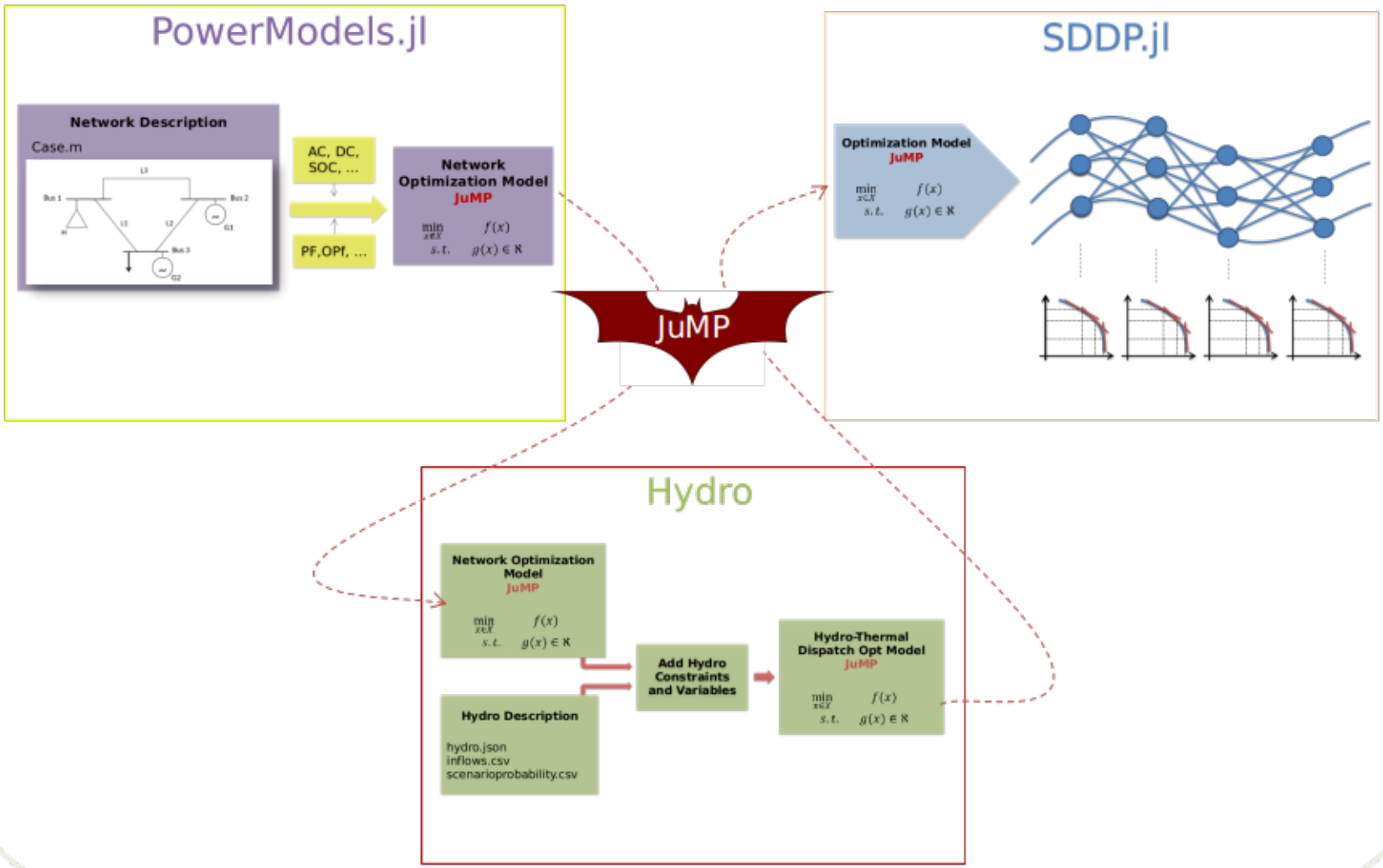
$$\min_{x \in X} \quad f(x)$$
$$s.t. \quad g(x) \in \aleph$$



Hydro


Network Optimization Model
JuMP
$\min_{x \in X} \quad f(x)$
$s.t. \quad g(x) \in \aleph$
Hydro Description
hydro.json
inflows.csv
scenarioprobability.csv
Add Hydro Constraints and Variables
Hydro-Thermal Dispatch Opt Model
JuMP
$\min_{x \in X} \quad f(x)$
$s.t. \quad g(x) \in \aleph$

HydroPowerModels.jl

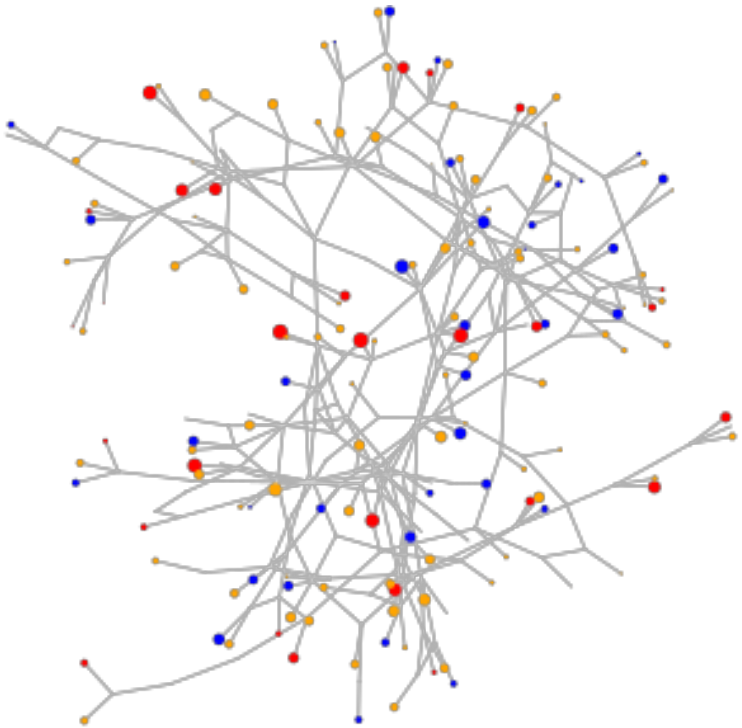


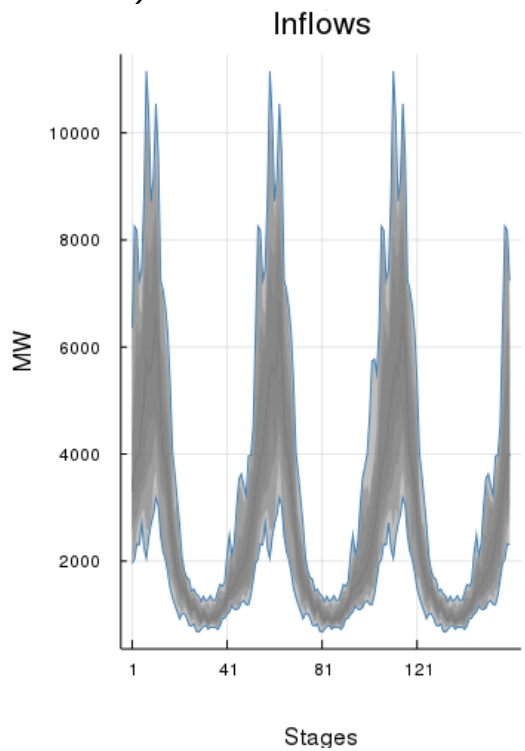
Example Case of a Realistic Hydrothermal Dispatch

Systems specifications:

- Number of buses: 166
- Number of generators: 145
- Number of loads: 286
- Number of branches: 235

In order for a qualitative view of the system, the package disposes a graph illustration plot (`plot_grid()`) and a scenarios quantile plot (`plotscenarios()`):





HydroPowerModels.jl Usage

- HydroPowerModels.jl gives you an interface to easily implement the described model.
- Once the case has been specified in the respective files (`PowerModels.json`, `hydro.json`, and `inflows.csv`) inside a case folder, a Hydrothermal Economic dispatch Operation may be planned:

First import the necessary packages:

```
> using HydroPowerModels
> using GLPK # one possible optimization solver
```

Load Case by passing the respective folder:

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

Dict{Any,Any} with 2 entries:
    "powersystem" => Dict{ ...
    "hydro"       => Dict{ ...
```

Use `create_param` to create a set of problem parameters:

```
> params = create_param(
      stages          = 160,
      stage_hours     = 168.0,
      model_constructor_grid = DCPPowerModel,
      optimizer       = with_optimizer(GLPK.Optimizer))
```

Build the Model and execute the SDDP method to optimize a policy:

```
> m = hydrothermaloperation(data, params);
> HydroPowerModels.train(m)
```

Finally, simulate the performance of the policy:

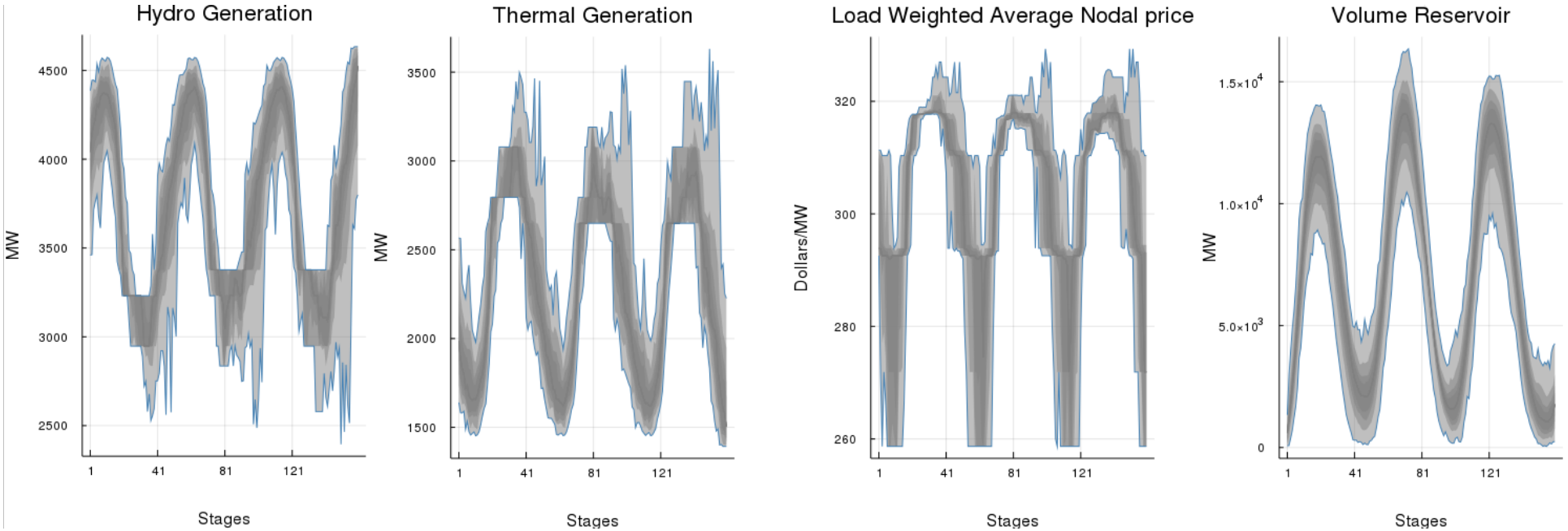
```
> results = HydroPowerModels.simulate(m, 1000);

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
```

Results Case

- Ploting results is easy! The function '`plot_aggregate_results()`' receives a results dictionary and generates the most common plots for a hydrothermal dispatch:

```
> plot_aggregated_results(results)
```



Documentation and More Examples

- Detailed Documentation about installation, usage and testing of the package can be found at: **Docs HydroPowerModels.jl**
- Under Examples in the documentation there are a few Jupyter like cases and results to help discussions and learning.

Other Packages

- This is one of the many open-source projects develop by LAMPS: **LAMPSPUC Github**

Bibliography

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

Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017. doi: 10.1137/15M1020575.

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. doi: 10.23919/PSCC.2018.8442948.

Oscar Dowson and Lea Kapelevich. SDDP.jl: a Julia package for Stochastic Dual Dynamic Programming. *Optimization Online*, 2017. URL `http://www.optimization-online.org/DB_HTML/2017/12/6388.html`.

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.

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.

SDDP.jl

```
# Model Definition
m = SDDPModel(
                sense   = :Min,
                stages  = params["stages"],
                solver  = params["solver"],
        objective_bound = 0.0
                                    ) do sp,t

    # build eletric grid model using PowerModels
    pm = PowerModels.build_generic_model(data["powersystem"], params["model_constructor_grid"],
        params["post_method"], jump_model=sp, setting = params["setting"])

    # create reference to variables
    createvarrefs(sp,pm)

    # save GenericPowerModel
    sp.ext[:pm] = pm

    # resevoir variables
    variable_volume(sp, data)

    # outflow and spillage variables
    variable_outflow(sp, data)
    variable_spillage(sp, data)

    # hydro balance
    variable_inflow(sp, data)
    rainfall_noises(sp, data, cidx(t,data["hydro"]["size_inflow"][1]))
    setnoiseprobability!(sp, data["hydro"]["scenario_probabilities"][cidx(t,data["hydro"]["size_inflow"][1]),:])
    constraint_hydro_balance(sp, data)

    # hydro_generation
    constraint_hydro_generation(sp, data, pm)

    # Stage objective
    @stageobjective(sp, sp.obj + sum(data["hydro"]["Hydrogenerators"][i]["spill_cost"]*sp[:spill][i] for i=1:data["hydro"]["nHyd"]))
end
```

PowerModels.jl

HydroPowerModels.jl