# Distributed Hydropower Models in StochasticPrograms.jl

Martin Biel

KTH - Royal Institute of Technology

July 30, 2019

# Motivation

- Simulation of hydro power operations for decision-support

# Motivation

- Simulation of hydro power operations for decision-support
  - Future electricity prices unknown
  - Irregular power production: solar and wind
  - Nuclear power phase-out

# Motivation

- Simulation of hydro power operations for decision-support
  - Future electricity prices unknown
  - Irregular power production: solar and wind
  - Nuclear power phase-out
- Common: trade-off between accuracy and computation time

# Motivation

- Simulation of hydro power operations for decision-support
  - Future electricity prices unknown
  - Irregular power production: solar and wind
  - Nuclear power phase-out
- Common: trade-off between accuracy and computation time
- Aim: **provide reliable decision-support in real time**

# Motivation

- Simulation of hydro power operations for decision-support
  - Future electricity prices unknown
  - Irregular power production: solar and wind
  - Nuclear power phase-out
- Common: trade-off between accuracy and computation time
- Aim: **provide reliable decision-support in real time**
  - Accurate models: optimal model reductions
  - Fast computations: scalable algorithms on commodity hardware

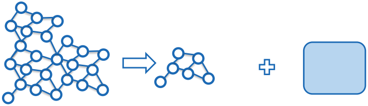


**Figure:** Manageable models.

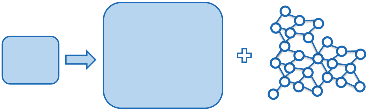


**Figure:** Scalable algorithms.

# Motivation

- Simulation of hydro power operations for decision-support
  - Future electricity prices unknown
  - Irregular power production: solar and wind
  - Nuclear power phase-out
- Common: trade-off between accuracy and computation time
- Aim: **provide reliable decision-support in real time**
  - Accurate models: optimal model reductions
  - Fast computations: **scalable algorithms on commodity hardware**

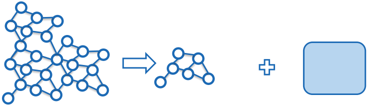


**Figure:** Manageable models.

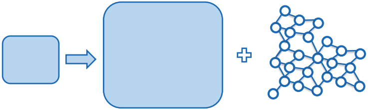


**Figure:** Scalable algorithms.

# Motivation

**Stochastic programming for hydro power operations**

- Optimal orders on the day-ahead market
- Maintenance scheduling
- Long-term investments
- Wind/solar uncertainties
- ...

# Motivation

**Stochastic programming for hydro power operations**

- Optimal orders on the day-ahead market
- Maintenance scheduling
- Long-term investments
- Wind/solar uncertainties
- ...

**In short**

- Accurate decision-support under uncertainty
- Variety of parallel decomposition schemes

# Contribution

- `StochasticPrograms.jl`: framework for stochastic programming

# Contribution

- `StochasticPrograms.jl`: framework for stochastic programming
- Formulate, solve and analyze stochastic models

# Contribution

- `StochasticPrograms.jl`: framework for stochastic programming
- Formulate, solve and analyze stochastic models
- A collection of structure-exploiting algorithms

# Contribution

- `StochasticPrograms.jl`: framework for stochastic programming
- Formulate, solve and analyze stochastic models
- A collection of structure-exploiting algorithms
- Distributed-memory implementation for large-scale models

# Contribution

- `StochasticPrograms.jl`: framework for stochastic programming
- Formulate, solve and analyze stochastic models
- A collection of structure-exploiting algorithms
- Distributed-memory implementation for large-scale models
- Collection of (hydroelectric) energy planning models

# Contribution

- `StochasticPrograms.jl`: framework for stochastic programming
- Formulate, solve and analyze stochastic models
- A collection of structure-exploiting algorithms
- Distributed-memory implementation for large-scale models
- Collection of (hydroelectric) energy planning models

*Efficiently model and solve stochastic problems using expressive syntax*

# Outline

- `StochasticPrograms.jl` showcase
- Real-world application: the day-ahead problem
- Numerical experiments
- Final remarks

# StochasticPrograms.jl

# StochasticPrograms.jl

- Flexible and expressive problem definition

# StochasticPrograms.jl

- Flexible and expressive problem definition
- Deferred model instantiation
- Scenario data injection

# StochasticPrograms.jl

- Flexible and expressive problem definition
- Deferred model instantiation
- Scenario data injection
- Variety of tools for analyzing models
  - VSS
  - EVPI
  - Confidence intervals
  - ...

# StochasticPrograms.jl

- Flexible and expressive problem definition
- Deferred model instantiation
- Scenario data injection
- Variety of tools for analyzing models
  - VSS
  - EVPI
  - Confidence intervals
  - ...
- Memory-distributed

# StochasticPrograms.jl

- Flexible and expressive problem definition
- Deferred model instantiation
- Scenario data injection
- Variety of tools for analyzing models
  - VSS
  - EVPI
  - Confidence intervals
  - ...
- Memory-distributed
- Minimize data passing
  - Lightweight sampler objects to generate scenario data
  - Lightweight model recipes to generate second stage problems

# StochasticPrograms.jl

- Flexible and expressive problem definition
- Deferred model instantiation
- Scenario data injection
- Variety of tools for analyzing models
    - VSS
    - EVPI
    - Confidence intervals
    - ...
- Memory-distributed
- Minimize data passing
    - Lightweight sampler objects to generate scenario data
    - Lightweight model recipes to generate second stage problems
- Interface to structure-exploiting (distributed) solver algorithms
    - L-shaped variants (`LShapedSolvers.jl`)
    - Progressive-hedging variants (`ProgressiveHedgingSolvers.jl`)

# StochasticPrograms.jl - Simple model

$$
\begin{aligned}
\underset{x_1, x_2 \in \mathbb{R}}{\text{minimize}} \quad & 100x_1 + 150x_2 + \mathbb{E}_\omega[Q(x_1, x_2, \xi)] \\
\text{s.t.} \quad & x_1 + x_2 \leq 120 \\
& x_1 \geq 40 \\
& x_2 \geq 20
\end{aligned}
$$

where

$$
\begin{aligned}
Q(x_1, x_2, \xi) = \min_{y_1, y_2 \in \mathbb{R}} \quad & q_1(\xi)y_1 + q_2(\xi)y_2 \\
\text{s.t.} \quad & 6y_1 + 10y_2 \leq 60x_1 \\
& 8y_1 + 5y_2 \leq 80x_2 \\
& 0 \leq y_1 \leq d_1(\xi) \\
& 0 \leq y_2 \leq d_2(\xi)
\end{aligned}
$$

# StochasticPrograms.jl - Simple model

```julia
simple_model = @stochastic_model begin
    @stage 1 begin
        @variable(model, x₁ >= 40)
        @variable(model, x₂ >= 20)
        @objective(model, Min, 100*x₁ + 150*x₂)
        @constraint(model, x₁ + x₂ <= 120)
    end
    @stage 2 begin
        @decision x₁ x₂
        @uncertain q₁ q₂ d₁ d₂
        @variable(model, 0 <= y₁ <= d₁)
        @variable(model, 0 <= y₂ <= d₂)
        @objective(model, Min, q₁*y₁ + q₂*y₂)
        @constraint(model, 6*y₁ + 10*y₂ <= 60*x₁)
        @constraint(model, 8*y₁ + 5*y₂ <= 80*x₂)
    end
end
```

# StochasticPrograms.jl - Simple model

```julia
simple_model = @stochastic_model begin
    @stage 1 begin
        @variable(model, x₁ >= 40)
        @variable(model, x₂ >= 20)
        @objective(model, Min, 100*x₁ + 150*x₂)
        @constraint(model, x₁ + x₂ <= 120)
    end
    @stage 2 begin
        @decision x₁ x₂
        @uncertain q₁ q₂ d₁ d₂
        @variable(model, 0 <= y₁ <= d₁)
        @variable(model, 0 <= y₂ <= d₂)
        @objective(model, Min, q₁*y₁ + q₂*y₂)
        @constraint(model, 6*y₁ + 10*y₂ <= 60*x₁)
        @constraint(model, 8*y₁ + 5*y₂ <= 80*x₂)
    end
end
```

JuMP syntax

# StochasticPrograms.jl - Simple model

```julia
simple_model = @stochastic_model begin
    @stage 1 begin
        @variable(model, x₁ >= 40)
        @variable(model, x₂ >= 20)
        @objective(model, Min, 100*x₁ + 150*x₂)
        @constraint(model, x₁ + x₂ <= 120)
    end
    @stage 2 begin
        @decision x₁ x₂
        @uncertain q₁ q₂ d₁ d₂
        @variable(model, 0 <= y₁ <= d₁)
        @variable(model, 0 <= y₂ <= d₂)
        @objective(model, Min, q₁*y₁ + q₂*y₂)
        @constraint(model, 6*y₁ + 10*y₂ <= 60*x₁)
        @constraint(model, 8*y₁ + 5*y₂ <= 80*x₂)
    end
end
```

$$\begin{aligned} \underset{x_1, x_2 \in \mathbb{R}}{\text{minimize}} \quad & 100x_1 + 150x_2 \\ \text{s.t.} \quad & x_1 + x_2 \leq 120 \\ & x_1 \geq 40 \\ & x_2 \geq 20 \end{aligned}$$

# StochasticPrograms.jl - Simple model

```
simple_model = @stochastic_model begin
    @stage 1 begin
        @variable(model, x₁ >= 40)
        @variable(model, x₂ >= 20)
        @objective(model, Min, 100*x₁ + 150*x₂)
        @constraint(model, x₁ + x₂ <= 120)
    end
    @stage 2 begin
        @decision x₁ x₂
        @uncertain q₁ q₂ d₁ d₂
        @variable(model, 0 <= y₁ <= d₁)
        @variable(model, 0 <= y₂ <= d₂)
        @objective(model, Min, q₁*y₁ + q₂*y₂)
        @constraint(model, 6*y₁ + 10*y₂ <= 60*x₁)
        @constraint(model, 8*y₁ + 5*y₂ <= 80*x₂)
    end
end
```

$$\begin{aligned} \underset{y_1, y_2 \in \mathbb{R}}{\text{minimize}} \quad & q_1(\xi)\, y_1 + q_2(\xi)\, y_2 \\ \text{s.t.} \quad & 6y_1 + 10y_2 \leq 60\, x_1 \\ & 8y_1 + 5y_2 \leq 80\, x_2 \\ & 0 \leq y_1 \leq d_1(\xi) \\ & 0 \leq y_2 \leq d_2(\xi) \end{aligned}$$

# StochasticPrograms.jl - Simple model

```
simple_model = @stochastic_model begin
    @stage 1 begin
        @variable(model, x₁ >= 40)
        @variable(model, x₂ >= 20)
        @objective(model, Min, 100*x₁ + 150*x₂)
        @constraint(model, x₁ + x₂ <= 120)
    end
    @stage 2 begin
        @decision x₁ x₂
        @uncertain q₁ q₂ d₁ d₂
        @variable(model, 0 <= y₁ <= d₁)
        @variable(model, 0 <= y₂ <= d₂)
        @objective(model, Min, q₁*y₁ + q₂*y₂)
        @constraint(model, 6*y₁ + 10*y₂ <= 60*x₁)
        @constraint(model, 8*y₁ + 5*y₂ <= 80*x₂)
    end
end
```

$$
\begin{aligned}
\underset{y_1, y_2 \in \mathbb{R}}{\text{minimize}} \quad & q_1(\xi)\, y_1 + q_2(\xi)\, y_2 \\
\text{s.t.} \quad & 6y_1 + 10y_2 \leq 60\, x_1 \\
& 8y_1 + 5y_2 \leq 80\, x_2 \\
& 0 \leq y_1 \leq d_1(\xi) \\
& 0 \leq y_2 \leq d_2(\xi)
\end{aligned}
$$

# StochasticPrograms.jl - Simple model

```julia
simple_model = @stochastic_model begin
    @stage 1 begin
        @variable(model, x₁ >= 40)
        @variable(model, x₂ >= 20)
        @objective(model, Min, 100*x₁ + 150*x₂)
        @constraint(model, x₁ + x₂ <= 120)
    end
    @stage 2 begin
        @decision x₁ x₂
        @uncertain q₁ q₂ d₁ d₂
        @variable(model, 0 <= y₁ <= d₁)
        @variable(model, 0 <= y₂ <= d₂)
        @objective(model, Min, q₁*y₁ + q₂*y₂)
        @constraint(model, 6*y₁ + 10*y₂ <= 60*x₁)
        @constraint(model, 8*y₁ + 5*y₂ <= 80*x₂)
    end
end
```

$$
\begin{aligned}
\underset{y_1, y_2 \in \mathbb{R}}{\text{minimize}} \quad & q_1(\xi)\, y_1 + q_2(\xi)\, y_2 \\
\text{s.t.} \quad & 6y_1 + 10y_2 \leq 60\, x_1 \\
& 8y_1 + 5y_2 \leq 80\, x_2 \\
& 0 \leq y_1 \leq d_1(\xi) \\
& 0 \leq y_2 \leq d_2(\xi)
\end{aligned}
$$

# StochasticPrograms.jl - Discrete distribution

```
s₁ = Scenario(q₁ = -24.0, q₂ = -28.0, d₁ = 500.0, d₂ = 100.0, probability = 0.4);
s₂ = Scenario(q₁ = -28.0, q₂ = -32.0, d₁ = 300.0, d₂ = 300.0, probability = 0.6);
simple_discrete = instantiate(simple_model, [s₁,s₂])

Stochastic program with:
 * 2 decision variables
 * 2 recourse variables
 * 2 scenarios of type Scenario
Solver is default solver
```

# StochasticPrograms.jl - Discrete distribution

```
print(simple_discrete)

 First-stage
==============
Min 100 x₁ + 150 x₂
Subject to
 x₁ + x₂ ≤ 120
 x₁ ≥ 40
 x₂ ≥ 20

 Second-stage
==============
Subproblem 1 (p = 0.40):
Min -24 y₁ - 28 y₂
Subject to
 -60 x₁ + 6 y₁ + 10 y₂ ≤ 0
 -80 x₂ + 8 y₁ + 5 y₂ ≤ 0
 0 ≤ y₁ ≤ 500
 0 ≤ y₂ ≤ 100

Subproblem 2 (p = 0.60):
Min -28 y₁ - 32 y₂
Subject to
 6 y₁ + 10 y₂ - 60 x₁ ≤ 0
 8 y₁ + 5 y₂ - 80 x₂ ≤ 0
 0 ≤ y₁ ≤ 300
 0 ≤ y₂ ≤ 300
```

# StochasticPrograms.jl - Discrete distribution

```
dep = DEP(simple_discrete)
print(dep)

Min 100 x₁ + 150 x₂ - 9.6 y₁₁ - 11.2 y₂₁ - 16.8 y₁₂ - 19.2 y₂₂
Subject to
 x₁ + x₂ ≤ 120
 6 y₁₁ + 10 y₂₁ - 60 x₁ ≤ 0
 8 y₁₁ + 5 y₂₁ - 80 x₂ ≤ 0
 6 y₁₂ + 10 y₂₂ - 60 x₁ ≤ 0
 8 y₁₂ + 5 y₂₂ - 80 x₂ ≤ 0
 x₁ ≥ 40
 x₂ ≥ 20
 0 ≤ y₁₁ ≤ 500
 0 ≤ y₂₁ ≤ 100
 0 ≤ y₁₂ ≤ 300
 0 ≤ y₂₂ ≤ 300
```

# StochasticPrograms.jl - Discrete distribution

```
vrp = VRP(simple_discrete, solver = glpk)
-855.83

vss = VSS(simple_discrete, solver = glpk)
286.92

evpi = EVPI(simple, solver = glpk)
662.92
```

# StochasticPrograms.jl - Continuous distribution

```
@sampler SimpleSampler = begin
    N::MvNormal

    SimpleSampler(μ, Σ) = new(MvNormal(μ, Σ))

    @sample Scenario begin
        x = rand(sampler.N)
        return Scenario(q₁ = x[1], q₂ = x[2], d₁ = x[3], d₂ = x[4])
    end
end

μ = [-28, -32, 300, 300]
Σ = [2 0.5 0 0
     0.5 1 0 0
     0 0 50 20
     0 0 20 30]

sampler = SimpleSampler(μ, Σ)
```

# StochasticPrograms.jl - Continuous distribution

```
saa = SAA(simple_model, sampler, 100)

Stochastic program with:
 * 2 decision variables
 * 2 recourse variables
 * 100 scenarios of type Scenario
Solver is default solver
```

# StochasticPrograms.jl - Continuous distribution

```
saa = SAA(simple_model, sampler, 100)

Stochastic program with:
 * 2 decision variables
 * 2 recourse variables
 * 100 scenarios of type Scenario
Solver is default solver
```

```
confidence_interval(simple_model, sampler; solver = glpk, confidence = 0.95,
        N = 100)
Confidence interval (p = 95%): [-2630.44 - -2389.31]
```

# StochasticPrograms.jl - Continuous distribution

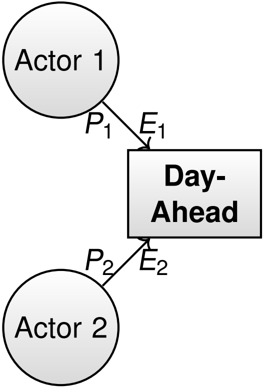
```
saa = SAA(simple_model, sampler, 100)

Stochastic program with:
 * 2 decision variables
 * 2 recourse variables
 * 100 scenarios of type Scenario
Solver is default solver
```

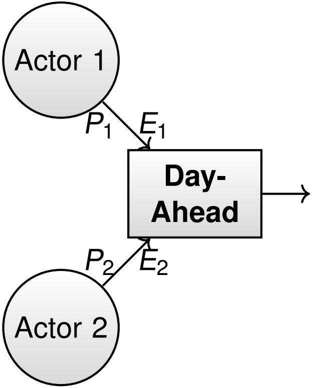
```
confidence_interval(simple_model, sampler; solver = glpk, confidence = 0.95,
        N = 100)
Confidence interval (p = 95%): [-2630.44 - -2389.31]
```

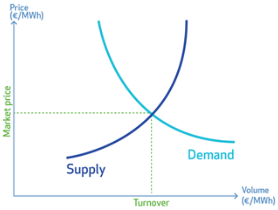
```
confidence_interval(simple_model, sampler; solver = glpk, confidence = 0.95,
        N = 1000)
Confidence interval (p = 95%): [-2568.90 - -2509.78]
```

# Day-ahead problem - Electricity market



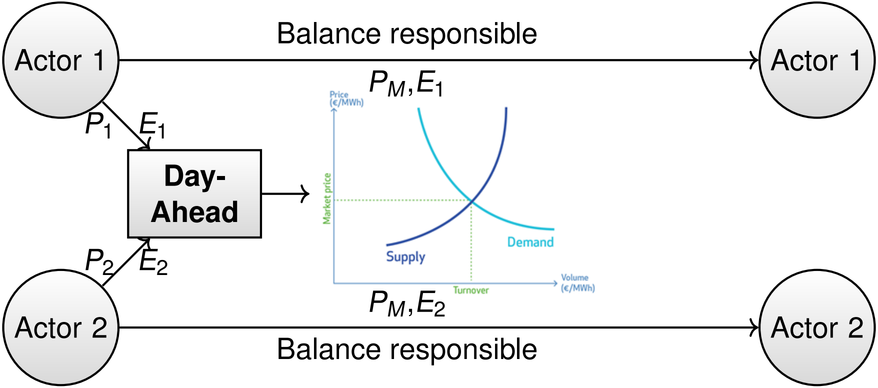
# Day-ahead problem - Electricity market



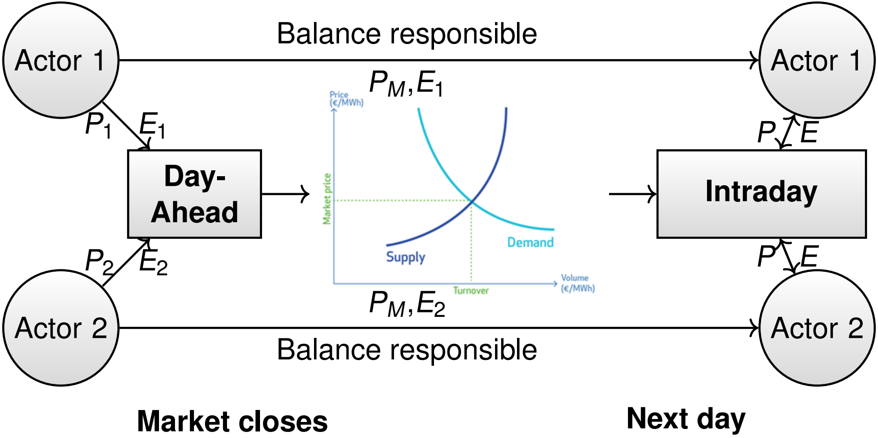

**Market closes**

# Day-ahead problem - Electricity market




**Market closes** **Next day**

# Day-ahead problem - Electricity market



# Day-ahead problem - Electricity market

## Order Types

- Single Hourly Order
  - Price independent
  - Price Dependent
- Block Order
  - Regular
  - Linked
- Exclusive Group
- Flexible Order

# Day-ahead problem - Electricity market

**Order Types**

- **Single Hourly Order**
  - **Price independent**
  - **Price Dependent**
- **Block Order**
  - **Regular**
  - Linked
- Exclusive Group
- Flexible Order

# Day-ahead problem - Single order

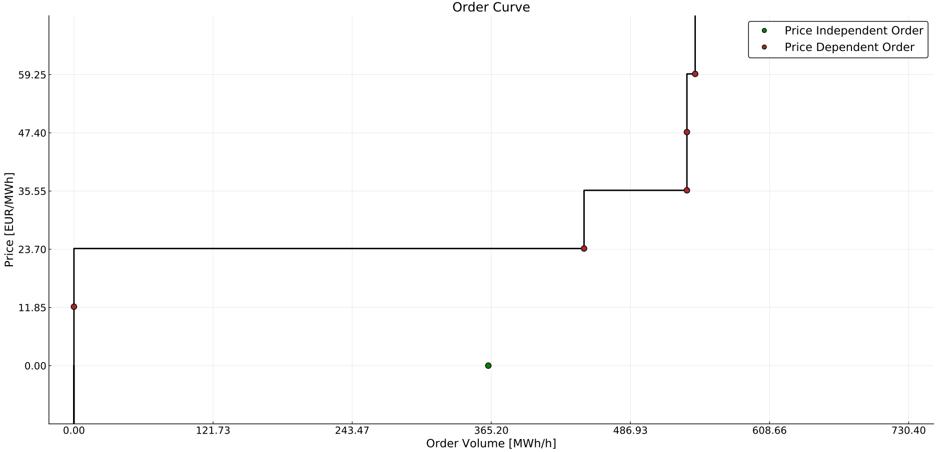



**Figure:** Single hourly order.

# Day-ahead problem - Single order

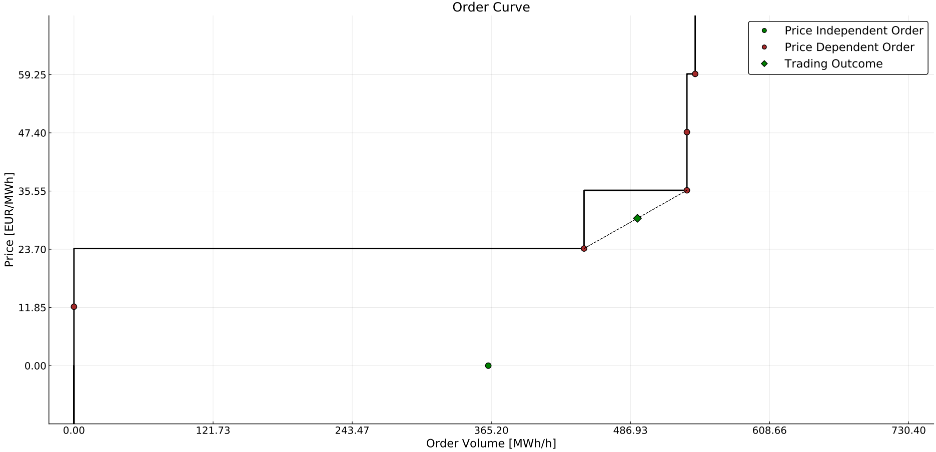



**Figure:** Interpolated energy volume for a given market price.

# Day-ahead problem - Block order

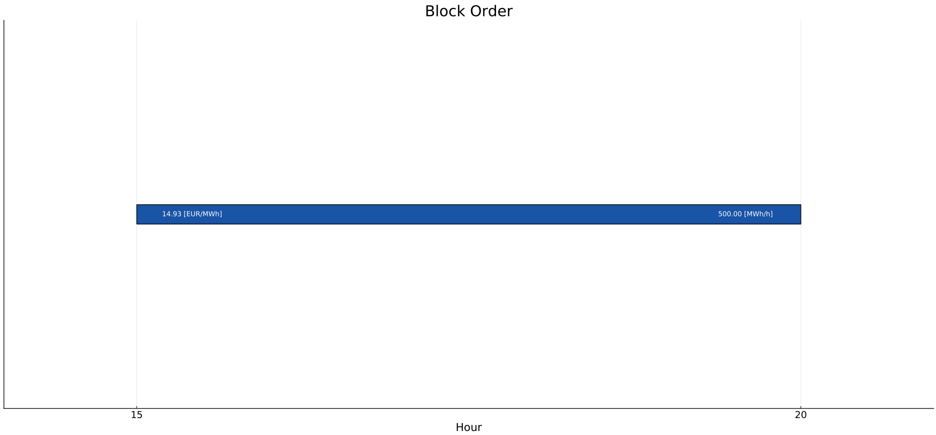



**Figure:** Block order between 15:00-20:00.

# Day-ahead problem - Block order

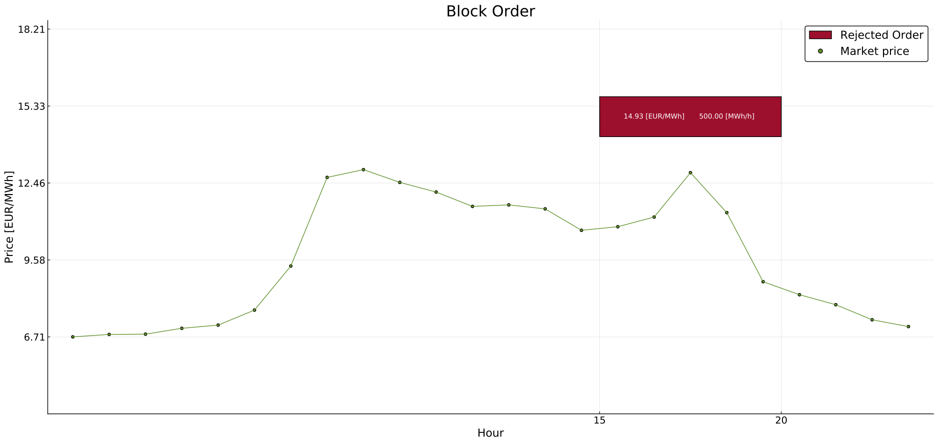



**Figure:** Rejected after market price settlement.

# Day-ahead problem - Block order


**Figure:** Accepted after market price settlement.

# Day-ahead problem - Setting

- Price taking hydropower producer trading in the NordPool market

# Day-ahead problem - Setting

- Price taking hydropower producer trading in the NordPool market
- All power stations in the Swedish river Skellefteälven

# Day-ahead problem - Setting

- Price taking hydropower producer trading in the NordPool market
- All power stations in the Swedish river Skellefteälven
- First stage: hourly electricity volume bids for the upcoming day
  - Single hourly orders
  - Block orders

# Day-ahead problem - Setting

- Price taking hydropower producer trading in the NordPool market
- All power stations in the Swedish river Skellefteälven
- First stage: hourly electricity volume bids for the upcoming day
  - Single hourly orders
  - Block orders
- Second stage: optimize day-ahead production
  - Bid dispatch after market price realization
  - Imbalances penalized in intraday market
  - Water flow conversation (including water travel time)
  - Maximize profits in the market *and the future value of water*

# Day-ahead problem - Setting

- Price taking hydropower producer trading in the NordPool market
- All power stations in the Swedish river Skellefteälven
- First stage: hourly electricity volume bids for the upcoming day
  - Single hourly orders
  - Block orders
- Second stage: optimize day-ahead production
  - Bid dispatch after market price realization
  - Imbalances penalized in intraday market
  - Water flow conversation (including water travel time)
  - Maximize profits in the market *and the future value of water*
- Full model defined in `HydroModels.jl`

# Day-ahead problem - Data

**Deterministic**

- Physical parameters for power plants in Skellefteälven
- Trade regulations from NordPool

**Uncertain**

- Day-ahead prices from NordPool
- Mean water flows in Skellefteälven from SMHI

# Day-ahead problem - Data

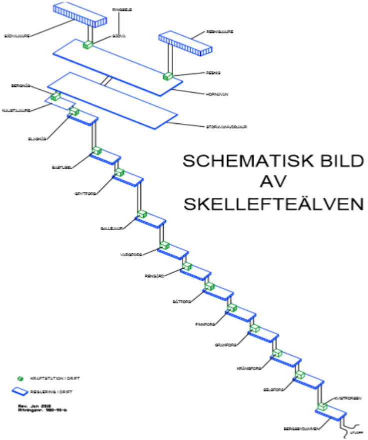



**Figure:** Schematic of the power stations in Skellefteälven.

# Day-ahead problem - Data

EUR/MWh

All hours are in CET/CEST. Last update: **Today 12:42 CET/CEST**.

| 29-07-2019 | SYS | SE1 | SE2 | SE3 | SE4 | FI | DK1 | DK2 | Oslo | Kr.sand | Bergen | Molde | Tr.heim | Tromsø | EE | LV | LT | AT | BE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 - 01 | 36,96 | 36,96 | 36,96 | 36,96 | 36,96 | 36,96 | 35,77 | 36,96 | 36,96 | 36,96 | 36,96 | 36,96 | 36,96 | 36,96 | 36,96 | 36,96 | 36,96 | 35,77 | 35,77 |
| 01 - 02 | 35,18 | 35,18 | 35,18 | 35,18 | 35,18 | 35,18 | 34,05 | 35,18 | 35,18 | 35,18 | 35,18 | 35,18 | 35,18 | 35,18 | 35,18 | 35,18 | 35,18 | 34,05 | 34,05 |
| 02 - 03 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 33,73 | 32,65 | 32,65 |
| 03 - 04 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 34,37 | 32,46 | 25,59 |
| 04 - 05 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 34,80 | 32,70 | 24,33 |
| 05 - 06 | 36,41 | 36,41 | 36,41 | 36,41 | 36,41 | 36,41 | 35,43 | 36,41 | 36,41 | 36,41 | 36,41 | 36,41 | 36,41 | 36,41 | 36,41 | 36,41 | 36,41 | 35,55 | 34,85 |
| 06 - 07 | 39,93 | 39,77 | 39,77 | 39,77 | 39,77 | 55,32 | 40,75 | 39,77 | 39,77 | 39,77 | 39,77 | 39,77 | 39,77 | 39,77 | 55,32 | 55,32 | 55,32 | 40,82 | 39,60 |
| 07 - 08 | 41,08 | 40,55 | 40,55 | 40,55 | 51,95 | 66,78 | 51,95 | 51,95 | 40,55 | 40,55 | 40,55 | 40,55 | 40,55 | 40,55 | 66,78 | 66,78 | 66,78 | 53,50 | 39,90 |
| 08 - 09 | 41,60 | 40,82 | 40,82 | 40,82 | 57,40 | 74,17 | 57,40 | 57,40 | 40,82 | 40,82 | 40,82 | 40,82 | 40,82 | 40,82 | 74,17 | 74,17 | 74,17 | 60,08 | 45,32 |
| 09 - 10 | 41,89 | 41,21 | 41,21 | 41,21 | 51,51 | 71,89 | 51,51 | 51,51 | 41,21 | 41,21 | 41,21 | 41,21 | 41,21 | 41,21 | 77,36 | 77,36 | 77,36 | 52,08 | 47,06 |
| 10 - 11 | 42,01 | 41,48 | 41,48 | 41,48 | 48,84 | 69,77 | 48,84 | 48,84 | 41,43 | 41,43 | 41,43 | 41,48 | 41,48 | 41,48 | 77,33 | 77,33 | 77,33 | 50,20 | 44,93 |
| 11 - 12 | 42,10 | 41,56 | 41,56 | 41,56 | 48,29 | 76,85 | 48,29 | 48,29 | 41,56 | 41,56 | 41,56 | 41,56 | 41,56 | 41,56 | 77,34 | 77,34 | 77,34 | 49,94 | 43,57 |
| 12 - 13 | 42,08 | 41,46 | 41,46 | 41,46 | 47,35 | 73,26 | 47,35 | 47,35 | 41,46 | 41,46 | 41,46 | 41,46 | 41,46 | 41,46 | 78,91 | 78,91 | 78,91 | 48,95 | 42,76 |
| 13 - 14 | 41,61 | 41,37 | 41,37 | 41,37 | 45,72 | 64,63 | 45,72 | 45,72 | 40,31 | 40,31 | 40,31 | 41,37 | 41,37 | 41,37 | 80,07 | 80,07 | 80,07 | 48,20 | 38,89 |
| 14 - 15 | 41,47 | 41,21 | 41,21 | 41,21 | 45,30 | 63,68 | 45,30 | 45,30 | 40,03 | 40,03 | 40,03 | 41,21 | 41,21 | 41,21 | 77,43 | 77,43 | 77,43 | 48,92 | 35,32 |
| 15 - 16 | 40,98 | 41,00 | 41,00 | 41,00 | 45,13 | 61,61 | 45,13 | 45,13 | 39,67 | 39,67 | 39,67 | 41,00 | 41,00 | 41,00 | 76,28 | 76,28 | 76,28 | 48,91 | 34,02 |
| 16 - 17 | 40,99 | 40,85 | 40,85 | 40,85 | 44,95 | 60,00 | 44,95 | 44,95 | 40,16 | 40,16 | 40,16 | 40,85 | 40,85 | 40,85 | 60,00 | 60,00 | 60,00 | 47,54 | 37,78 |
| 17 - 18 | 41,19 | 40,92 | 40,92 | 40,92 | 45,21 | 56,05 | 45,21 | 45,21 | 40,92 | 40,92 | 40,92 | 40,92 | 40,92 | 40,92 | 56,05 | 56,05 | 56,05 | 45,21 | 45,21 |
| 18 - 19 | 41,51 | 41,05 | 41,05 | 41,05 | 49,50 | 60,09 | 49,50 | 49,50 | 41,05 | 41,05 | 41,05 | 41,05 | 41,05 | 41,05 | 60,09 | 60,09 | 60,09 | 49,50 | 48,05 |
| 19 - 20 | 41,27 | 40,81 | 40,81 | 40,81 | 60,74 | 60,67 | 60,74 | 60,74 | 40,81 | 40,81 | 40,81 | 40,81 | 40,81 | 40,81 | 60,74 | 60,74 | 60,74 | 58,22 | 52,99 |
| 20 - 21 | 40,95 | 40,69 | 40,69 | 40,69 | 56,07 | 55,36 | 60,50 | 60,50 | 40,69 | 40,69 | 40,69 | 40,69 | 40,69 | 40,69 | 56,07 | 56,07 | 56,07 | 57,80 | 52,17 |
| 21 - 22 | 40,45 | 40,39 | 40,39 | 40,39 | 51,96 | 51,96 | 53,23 | 53,23 | 40,39 | 40,39 | 40,39 | 40,39 | 40,39 | 40,39 | 51,96 | 51,96 | 51,96 | 51,90 | 49,12 |
| 22 - 23 | 39,99 | 40,08 | 40,08 | 40,08 | 43,02 | 43,02 | 49,38 | 49,38 | 40,08 | 40,08 | 40,08 | 40,08 | 40,08 | 40,08 | 43,02 | 43,02 | 43,02 | 49,38 | 49,38 |
| 23 - 00 | 39,32 | 39,39 | 39,39 | 39,39 | 39,39 | 39,39 | 43,25 | 43,25 | 39,39 | 39,39 | 39,39 | 39,39 | 39,39 | 39,39 | 39,39 | 39,39 | 39,39 | 43,25 | 43,25 |

**Figure:** Historical day-ahead prices 2013-2018 from NordPool.

# Day-ahead problem - Data

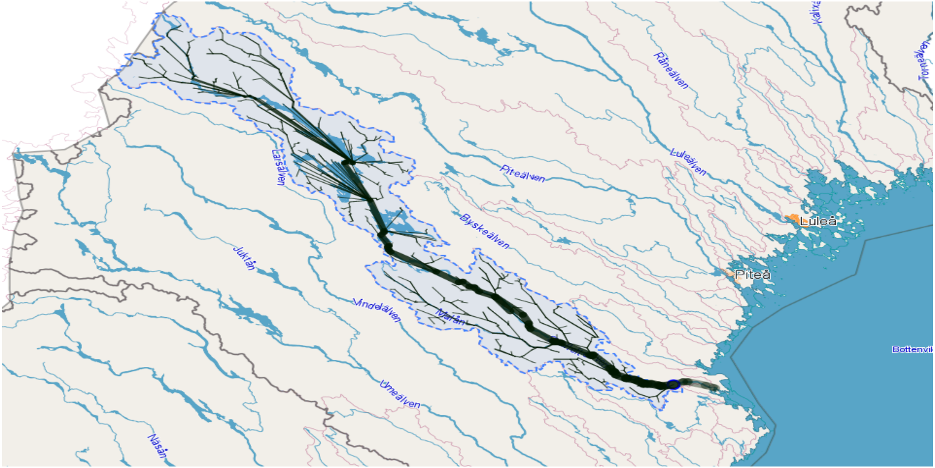



**Figure:** Mean water flow in Skellefteälven 1999-2018 from SMHI.

# Day-ahead problem - Forecasts

- Recurrent neural networks (GRU)

# Day-ahead problem - Forecasts

- Recurrent neural networks (GRU)
- Trained on price data and mean flow data separately

# Day-ahead problem - Forecasts

- Recurrent neural networks (GRU)
- Trained on price data and mean flow data separately
- Early stopping to prevent overfitting

# Day-ahead problem - Forecasts

- Recurrent neural networks (GRU)
- Trained on price data and mean flow data separately
- Early stopping to prevent overfitting
- Seasonality modeled through separate inputs to the network

# Day-ahead problem - Forecasts

- Recurrent neural networks (GRU)
- Trained on price data and mean flow data separately
- Early stopping to prevent overfitting
- Seasonality modeled through separate inputs to the network
- Driven by Gaussian noise

# Day-ahead problem - Forecasts

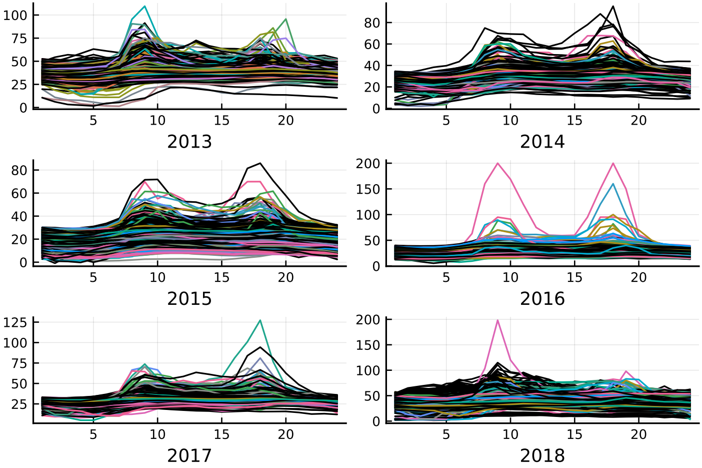



**Figure:** Price forecasts (black) and raw data (colored).

# Day-ahead problem - Forecasts

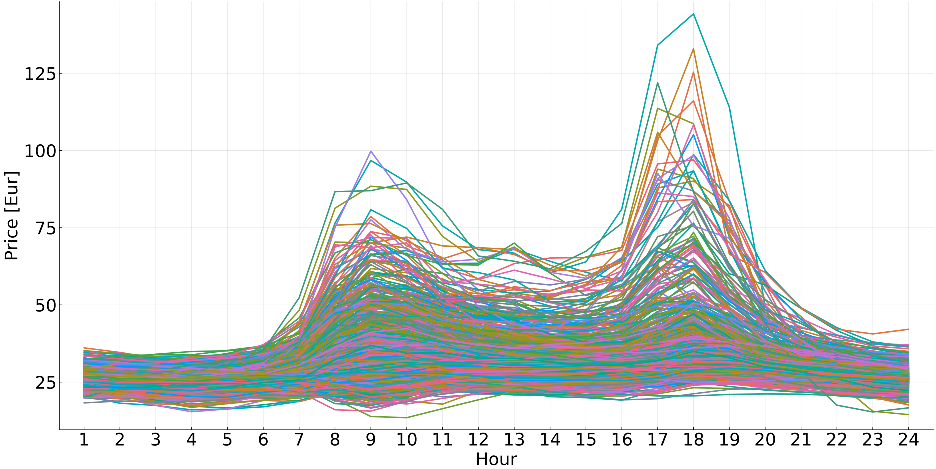



**Figure:** 1000 sampled price curves using RNN.

# Day-ahead problem - Forecasts

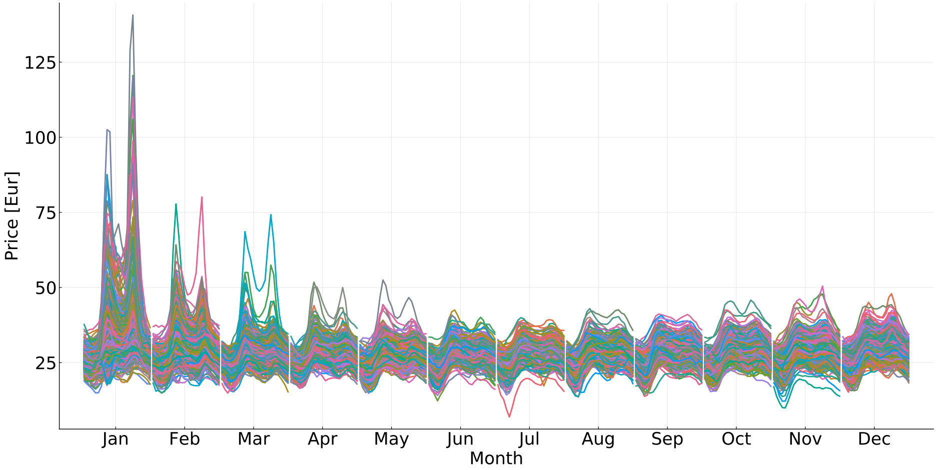



**Figure:** Price forecasts throughout a year.

# Day-ahead problem - Forecasts

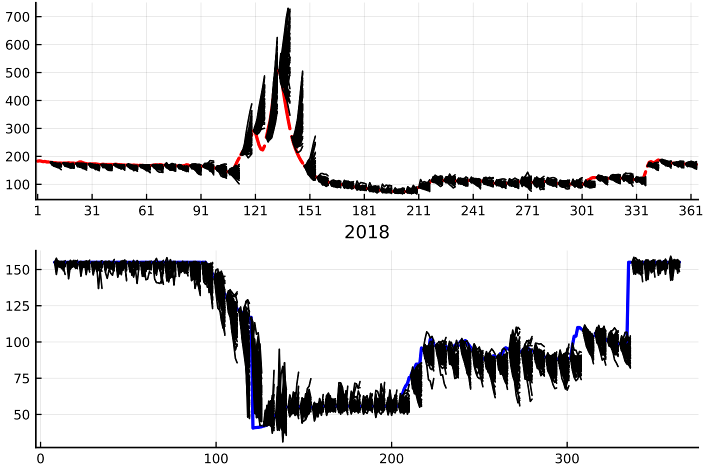



**Figure:** Mean flow forecasts (black) and raw data (colored).

# Day-ahead problem - Sampler

```
@scenario DayAheadScenario = begin
    ρ::PriceCurve{Float64}
    Q:Vector{Float64}
end
@sampler RecurrentDayAheadSampler = begin
    date::Date
    price_forecaster::Forecaster{:price}
    flow_forecaster::Forecaster{:flow}

    @sample DayAheadScenario begin
        prices = forecast(sampler.price_forecaster, month(sampler.date))
        flows = forecast(sampler.flow_forecaster, week(sampler.date))
        return DayAheadScenario(PriceCurve(prices), flows[1])
    end
end
```

# Day-ahead problem - Bidlevels

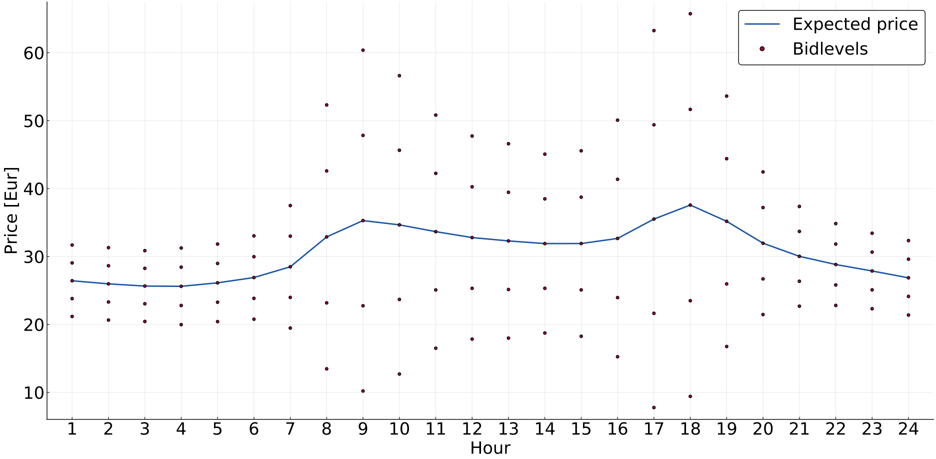



**Figure:** Available price points for bidding.

# Day-ahead problem - Value of water

- Marginal value of water has large impact on optimal dispatch

# Day-ahead problem - Value of water

- Marginal value of water has large impact on optimal dispatch
- Sometimes optimal to accept imbalance penalty and save water

# Day-ahead problem - Value of water

- Marginal value of water has large impact on optimal dispatch
- Sometimes optimal to accept imbalance penalty and save water
- Naive approach: production from excess water solved at mean price

# Day-ahead problem - Value of water

- Marginal value of water has large impact on optimal dispatch
- Sometimes optimal to accept imbalance penalty and save water
- Naive approach: production from excess water solved at mean price
- Leads to crude order strategies

# Day-ahead problem - Value of water

- Solve a dummy stochastic program:
  - First stage: water content in reservoirs
  - Second stage: optimize production over the coming week
  - Future prices and water inflows are uncertain

# Day-ahead problem - Value of water

- Solve a dummy stochastic program:
  - First stage: water content in reservoirs
  - Second stage: optimize production over the coming week
  - Future prices and water inflows are uncertain
- L-shaped generates a polyhedral objective approximation

# Day-ahead problem - Value of water

- Solve a dummy stochastic program:
  - First stage: water content in reservoirs
  - Second stage: optimize production over the coming week
  - Future prices and water inflows are uncertain
- L-shaped generates a polyhedral objective approximation
- Approximation used to model the expected future value of water

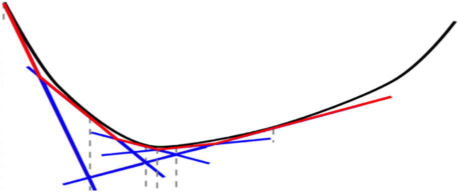


**Figure:** Polyhedral approximation.

# Day-ahead problem - Model

```julia
@stage 1 begin
    @parameters begin
        horizon = horizon
        indices = indices
        data = data
    end
    @unpack hours, plants, bids, blockbids, blocks = indices
    @unpack hydrodata, regulations = data
    # Variables
    # ==================================================
    @variable(model, xt_i[t = hours] >= 0)
    @variable(model, xt_d[i = bids, t = hours] >= 0)
    @variable(model, xb[i = blockbids, b = blocks] >= 0)
    ...
end
```

# Day-ahead problem - Model

```
@stage 2 begin
    ...
    @uncertain ρ, V from ξ::DayAheadScenario
    @decision xt_i xt_d xb
    # ------------------------------------------------------
    @variable(model, yt[t = hours] >= 0) # Hourly dispatch
    @variable(model, yb[b = blocks] >= 0) # Block dispatch
    @variable(model, z_up[t = hours] >= 0) # Power bought from intraday
    @variable(model, z_do[t = hours] >= 0) # Power sold to intraday
    @variable(model, 0 <= Q[p = plants, t = hours] <= Q̄) # Water discharge
    @variable(model, S[p = plants, t = hours] >= 0) # Spillage
    @variable(model, Qf[p = plants, t = hours] >= 0) # Incoming discharge
    @variable(model, Sf[p = plants, t = hours] >= 0) # Incoming spillage
    @variable(model, 0 <= M[p = plants, t = hours] <= M̄ # Reservoir content
    @variable(model, H[t = hours] >= 0) # Power production
    ...
    @objective(model, Max, net_profit + value_of_stored_water)
    ...
```

# Day-ahead problem - Model

```
...
# Bid-dispatch links
@constraint(model, hourlybids[t = hours],
            yt[t] == interpolate(ρ[t], bidlevels, xt_d[t]) + xt_i[t]
            )
@constraint(model, bidblocks[b = blocks],
                yb[b] == sum(xb[j,b] for j = accepted_blocks(b))
            )
# Hydrological balance
@constraint(model, hydro_constraints[p = plants, t = hours],
            # Previous reservoir content
            M[p,t] == (t > 1 ? M[p,t-1] : M₀[p])
            # Inflow
            + sum(Qf[i,t]+Sf[i,t] for i = upstream_plants[p])
            # Local inflow
            + V[p]
            # Outflow
            - (Q[p,t] + S[p,t])
            )
...
```

# Day-ahead problem - Model

```
    ...
    # Production
    @constraint(model, production[t = hours],
                H[t] == sum(hydrodata[p].μ[s]*Q[p,s,t]
                            for p = plants, s = segments)
                )
    # Load balance
    @constraint(model, loadbalance[t = hours],
                yt[t] + sum(yb[b] for b = blocks[t]) - H[t] == z_up[t] - z_do[t]
                )
    ...
    # Water travel time
    ...
    # Water value
    @constraint(model, water_value_approximation[c = 1:ncuts(water_value)],
                sum(water_value[c][p]*M[p,nhours(horizon)]
                    for p in plants)
                + sum(W[i]
                      for i in cut_indices(water_value[c]))
                >= cut_lb(water_value[c]))
end
```

# Day-ahead problem - Algorithm

- VSS typically low in day-ahead problems

# Day-ahead problem - Algorithm

- VSS typically low in day-ahead problems
- Generate tight confidence intervals trough sequential SAA algorithm

# Day-ahead problem - Algorithm

- VSS typically low in day-ahead problems
- Generate tight confidence intervals trough sequential SAA algorithm
- Ensure statistically significant VSS

# Day-ahead problem - Algorithm

- VSS typically low in day-ahead problems
- Generate tight confidence intervals trough sequential SAA algorithm
- Ensure statistically significant VSS
- SAA instances of ~2000 scenarios required to reach this bound
    - ~5 million variables
    - ~3.3 million constraints

# Day-ahead problem - Algorithm

**Sequential SAA**

- Lower bound: solve $M$ SAA models of size $N$
- Upper bound: decision evaluation on $T$ SAA models of size $\tilde{N} > N$
- Increase $N$ iteratively until confidence interval is tight enough

# Day-ahead problem - Algorithm

## Sequential SAA

- Lower bound: solve $M$ SAA models of size $N$
- Upper bound: decision evaluation on $T$ SAA models of size $\tilde{N} > N$
- Increase $N$ iteratively until confidence interval is tight enough

## Distributed L-shaped

- Regularization
  - Trust-regions
  - Level-sets
  - ...
- Aggregation
  - Static
  - Dynamic
  - Clustering
  - ...

# Results - Benchmarks

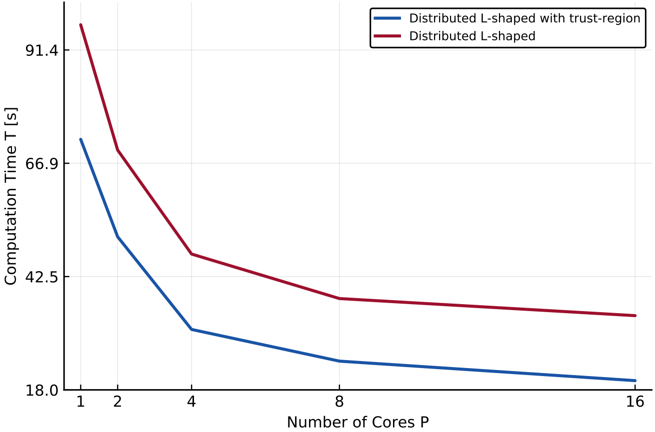



**Figure:** Distributed L-shaped performance.

# Results - Benchmarks

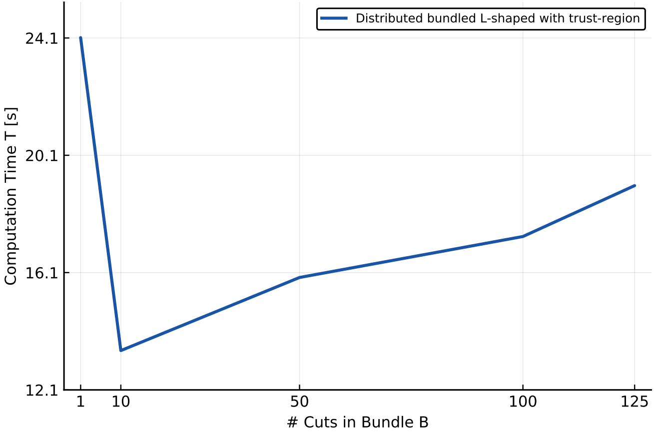



**Figure:** Distributed L-shaped performance using aggregation.

# Results - Day-ahead

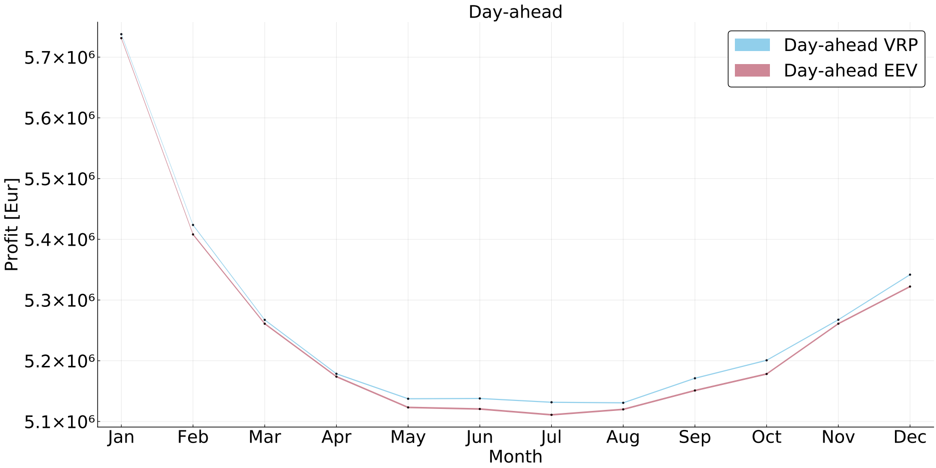



**Figure:** Day-ahead profits.

# Results - Day-ahead

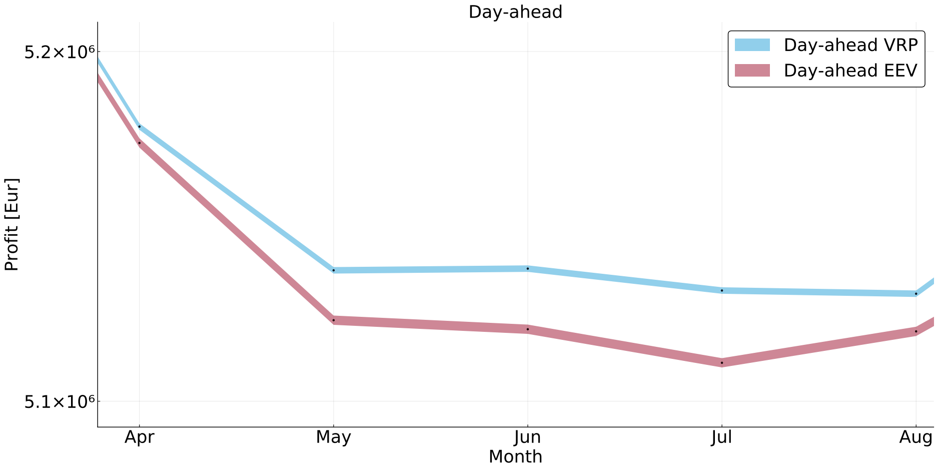



**Figure:** Day-ahead profits.

# Results - Day-ahead

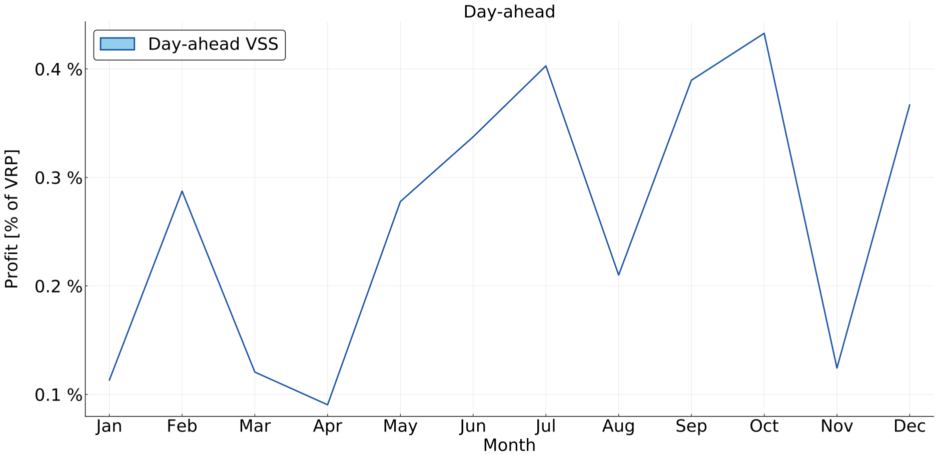



**Figure:** Day-ahead value of stochastic solution.

# Results - Day-ahead

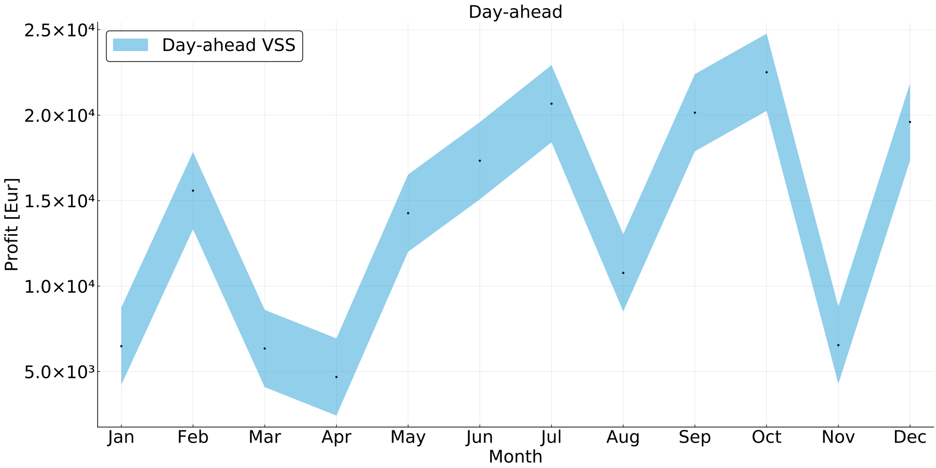



**Figure:** Day-ahead value of stochastic solution.

# Results - Order strategies


**Figure:** Day-ahead order strategy.

# Results - Order strategies

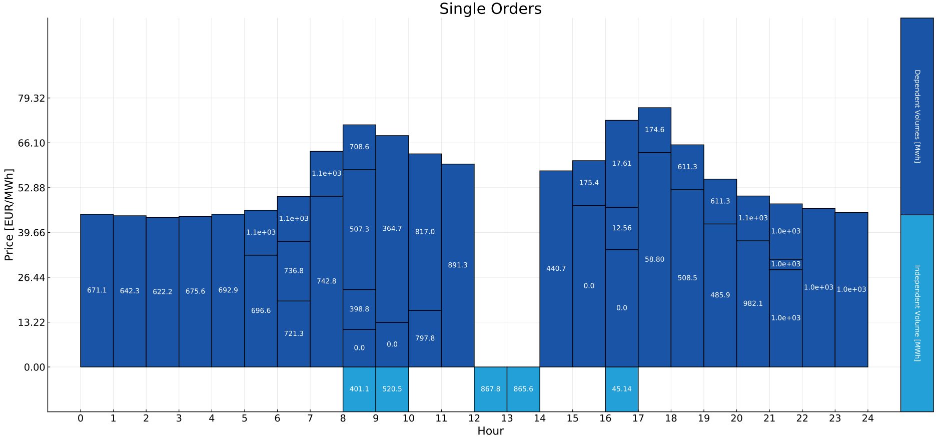



**Figure:** Day-ahead single hourly order strategy.

# Results - Order strategies


**Figure:** Result of single order strategy after realized market price.

# Results - Order strategies


**Figure:** Result of complete order strategy after realized market price.

# Results - Order strategies

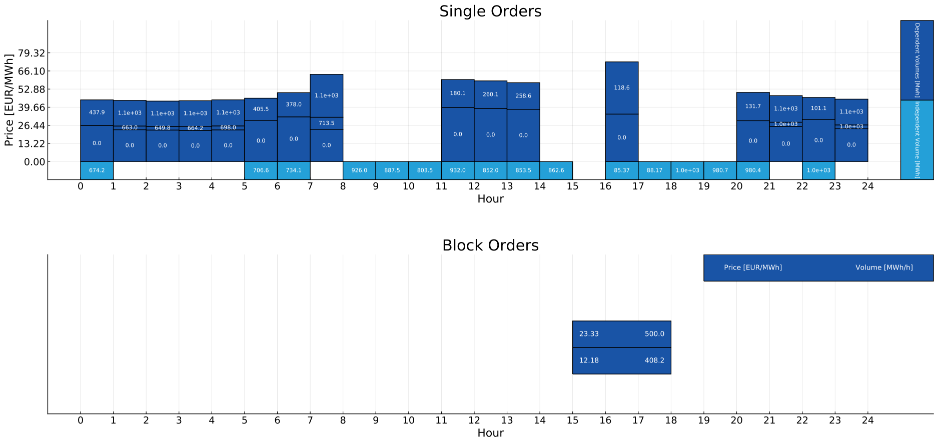



**Figure:** Deterministic order strategy.

# Final Remarks

### Discussion

- VSS linked to imbalance penalties in the intraday market

# Final Remarks

### Discussion

- VSS linked to imbalance penalties in the intraday market
- Results are only as accurate/useful as the water valuation

# Final Remarks

### Discussion

- VSS linked to imbalance penalties in the intraday market
- Results are only as accurate/useful as the water valuation
- Model improvements required.

# Final Remarks

## Discussion

- VSS linked to imbalance penalties in the intraday market
- Results are only as accurate/useful as the water valuation
- Model improvements required.
- Proof of concept for large-scale models in `StochasticPrograms.jl`

# Final Remarks

**Discussion**

- VSS linked to imbalance penalties in the intraday market
- Results are only as accurate/useful as the water valuation
- Model improvements required.
- Proof of concept for large-scale models in `StochasticPrograms.jl`

**Outlook on future/ongoing work**

- Formulate new energy planning models in `StochasticPrograms.jl`

# Final Remarks

**Discussion**

- VSS linked to imbalance penalties in the intraday market
- Results are only as accurate/useful as the water valuation
- Model improvements required.
- Proof of concept for large-scale models in `StochasticPrograms.jl`

**Outlook on future/ongoing work**

- Formulate new energy planning models in `StochasticPrograms.jl`
- Evaluate different aggregation schemes

# Final Remarks

**Discussion**

- VSS linked to imbalance penalties in the intraday market
- Results are only as accurate/useful as the water valuation
- Model improvements required.
- Proof of concept for large-scale models in `StochasticPrograms.jl`

**Outlook on future/ongoing work**

- Formulate new energy planning models in `StochasticPrograms.jl`
- Evaluate different aggregation schemes
- Sample-based algorithms as an alternative to sequential SAA

# Final Remarks

## Summary

- `StochasticPrograms.jl`: framework for stochastic programming

# Final Remarks

**Summary**

- `StochasticPrograms.jl`: framework for stochastic programming
- Large-scale day-ahead problem solved on compute cluster

# Final Remarks

## Summary

- `StochasticPrograms.jl`: framework for stochastic programming
- Large-scale day-ahead problem solved on compute cluster
- Tight confidence intervals through sequential SAA

# Final Remarks

## Summary

- `StochasticPrograms.jl`: framework for stochastic programming
- Large-scale day-ahead problem solved on compute cluster
- Tight confidence intervals through sequential SAA
- `StochasticPrograms.jl` is a registered Julia package

# Final Remarks

## Summary

- `StochasticPrograms.jl`: framework for stochastic programming
- Large-scale day-ahead problem solved on compute cluster
- Tight confidence intervals through sequential SAA
- `StochasticPrograms.jl` is a registered Julia package
- The full framework is open-source and freely available on Github

https://github.com/martinbiel