Optimal Day-Ahead Orders Using Stochastic Programming and Noise-Driven Recurrent Neural Networks

Martin Biel

KTH - Royal Institute of Technology

June 29, 2021

Contribution

Contribution

Motivation

- Determine optimal order strategies in a deregulated electricity market

Contribution

Motivation

- Determine optimal order strategies in a deregulated electricity market

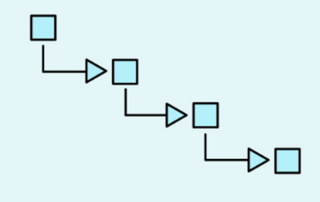
- Hydroelectric power production

Contribution

Motivation

- Determine optimal order strategies in a deregulated electricity market

- Hydroelectric power production
- Spatial dependence



Contribution

Motivation

- Determine optimal order strategies in a deregulated electricity market

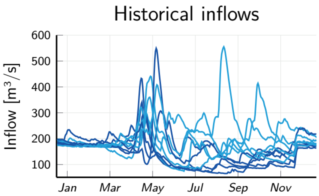
- Hydroelectric power production
- Spatial dependence
- Temporal dependence

Contribution

Motivation

- Determine optimal order strategies in a deregulated electricity market

- Uncertain local inflow

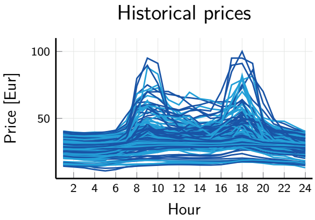


Contribution

Motivation

- Determine optimal order strategies in a deregulated electricity market

- Uncertain local inflow
- Uncertain electricity price

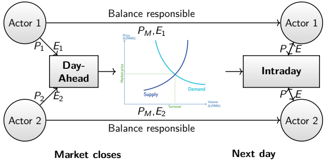


Contribution

Motivation

- Determine optimal order strategies in a deregulated electricity market

- Uncertain local inflow
- Uncertain electricity price
- Day-ahead markets

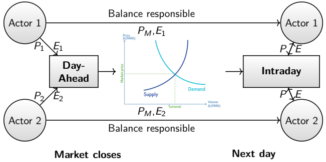


Contribution

Motivation

- Determine optimal order strategies in a deregulated electricity market

- Uncertain local inflow
- Uncertain electricity price
- Day-ahead markets



Contribution

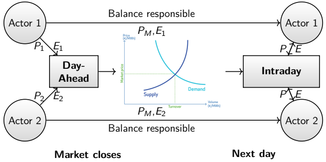
- Complete modeling procedure
  - Data gathering
  - Forecast generation
  - Model formulation
  - Optimization
  - Result visualization

Contribution

Motivation

- Determine optimal order strategies in a deregulated electricity market

- Uncertain local inflow
- Uncertain electricity price
- Day-ahead markets



Contribution

- Complete modeling procedure
  - Data gathering
  - Forecast generation
  - Model formulation
  - Optimization
  - Result visualization

Martin Biel. Optimal day-ahead orders using stochastic programming and noise-driven RNNs. *arXiv preprint arXiv:1910.04510*, 2019

Forecast generation

- Noise-driven recurrent neural network

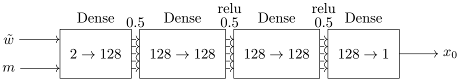


Figure: Initializer network in the price forecaster.

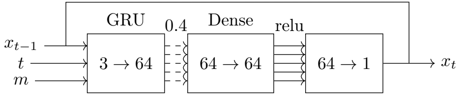


Figure: Sequence generation network in the price forecaster.

Forecast generation

- Noise-driven recurrent neural network
- Trained on price data and inflow data separately

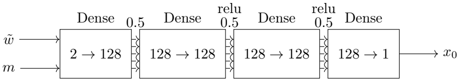


Figure: Initializer network in the price forecaster.

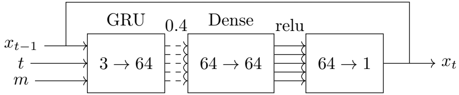


Figure: Sequence generation network in the price forecaster.

Forecast generation

- Noise-driven recurrent neural network
- Trained on price data and inflow data separately
- Seasonality modeled through separate inputs to the network

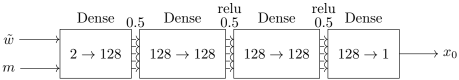


Figure: Initializer network in the price forecaster.

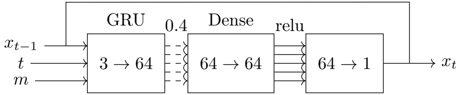


Figure: Sequence generation network in the price forecaster.

Forecast generation

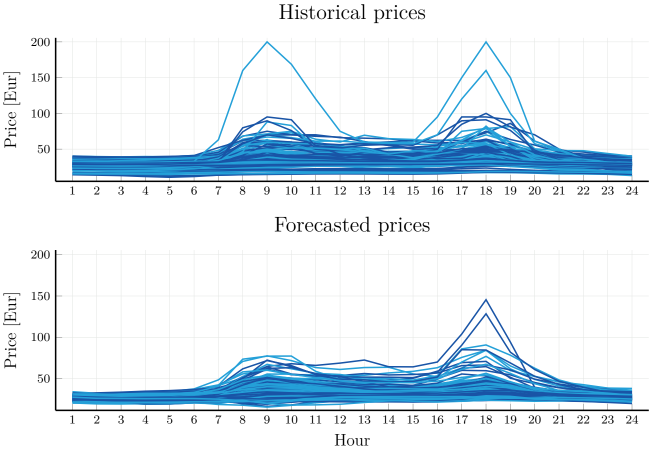


Figure: Historical electricity price curves in January and electricity price curves generated using the RNN forecaster in the same period.

Forecast generation

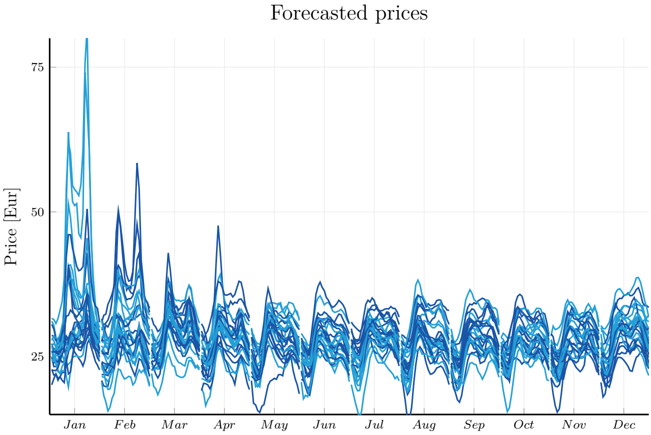


Figure: Daily electricity price curves predicted by the RNN forecaster in every month of the year.

Model formulation

- `StochasticPrograms.jl`: framework for stochastic programming

Model formulation

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

Model formulation

- `StochasticPrograms.jl`: framework for stochastic programming
- Formulate, solve and analyze stochastic models
- Distributed-memory implementation for large-scale models

Model formulation

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

Model formulation - StochasticPrograms.jl

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

Model formulation - StochasticPrograms.jl

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

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

Model formulation - StochasticPrograms.jl

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

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

Model formulation - Day-ahead problem

Full model details:

- Preprint: Martin Biel. Optimal day-ahead orders using stochastic programming and noise-driven RNNs.
  arXiv preprint arXiv:1910.04510, 2019
- Github: `github.com/martinbiel/HydroModels`

Optimization

- Generate tight confidence intervals using *sample average approximation*

Optimization

- Generate tight confidence intervals using *sample average approximation*
- Ensure a statistically significant *value of the stochastic solution*

Optimization

- Generate tight confidence intervals using *sample average approximation*
- Ensure a statistically significant *value of the stochastic solution*
- Sampled instances of ∼2000 scenarios required to reach this bound
  - ∼5 million variables
  - ∼3.3 million constraints

Optimization

- Generate tight confidence intervals using *sample average approximation*
- Ensure a statistically significant *value of the stochastic solution*
- Sampled instances of ~2000 scenarios required to reach this bound
  - ~5 million variables
  - ~3.3 million constraints
- Leverage distributed capabilities of `StochasticPrograms.jl`

Results

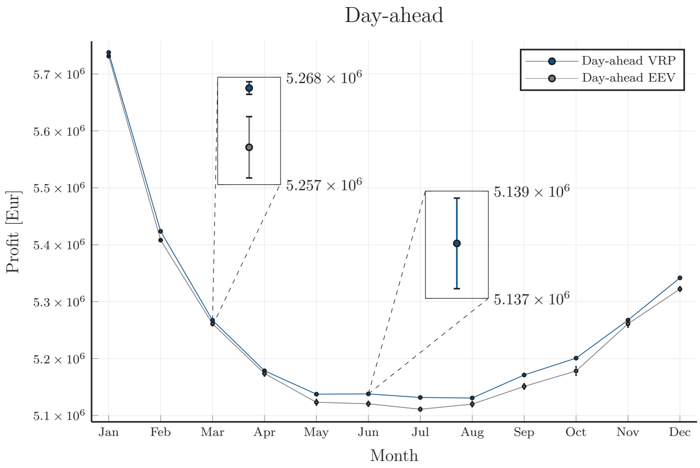


Figure: Seasonal variation of day-ahead VRP and EEV, including 95% confidence intervals.

Final remarks

Summary

- Large-scale day-ahead problems solved on a compute cluster

Final remarks

Summary

- Large-scale day-ahead problems solved on a compute cluster
- Noise-driven recurrent neural networks to sample scenarios

Final remarks

Summary

- Large-scale day-ahead problems solved on a compute cluster
- Noise-driven recurrent neural networks to sample scenarios
- Tight confidence intervals from SAA

Final remarks

Summary

- Large-scale day-ahead problems solved on a compute cluster
- Noise-driven recurrent neural networks to sample scenarios
- Tight confidence intervals from SAA
- Statistically significant VSS

Final remarks

Summary

- Large-scale day-ahead problems solved on a compute cluster
- Noise-driven recurrent neural networks to sample scenarios
- Tight confidence intervals from SAA
- Statistically significant VSS
- Proof of concept for large-scale models in `StochasticPrograms.jl`

Final remarks

Summary

- Large-scale day-ahead problems solved on a compute cluster
- Noise-driven recurrent neural networks to sample scenarios
- Tight confidence intervals from SAA
- Statistically significant VSS
- Proof of concept for large-scale models in `StochasticPrograms.jl`

Further information

- Contact: `mbiel@kth.se`
- Github: `github.com/martinbiel/StochasticPrograms.jl`
- Full paper