ApplicationDrivenLearning.jl

A Closed-Loop Prediction and Optimization Approach

Joaquim Dias Garcia (Soma Energy)

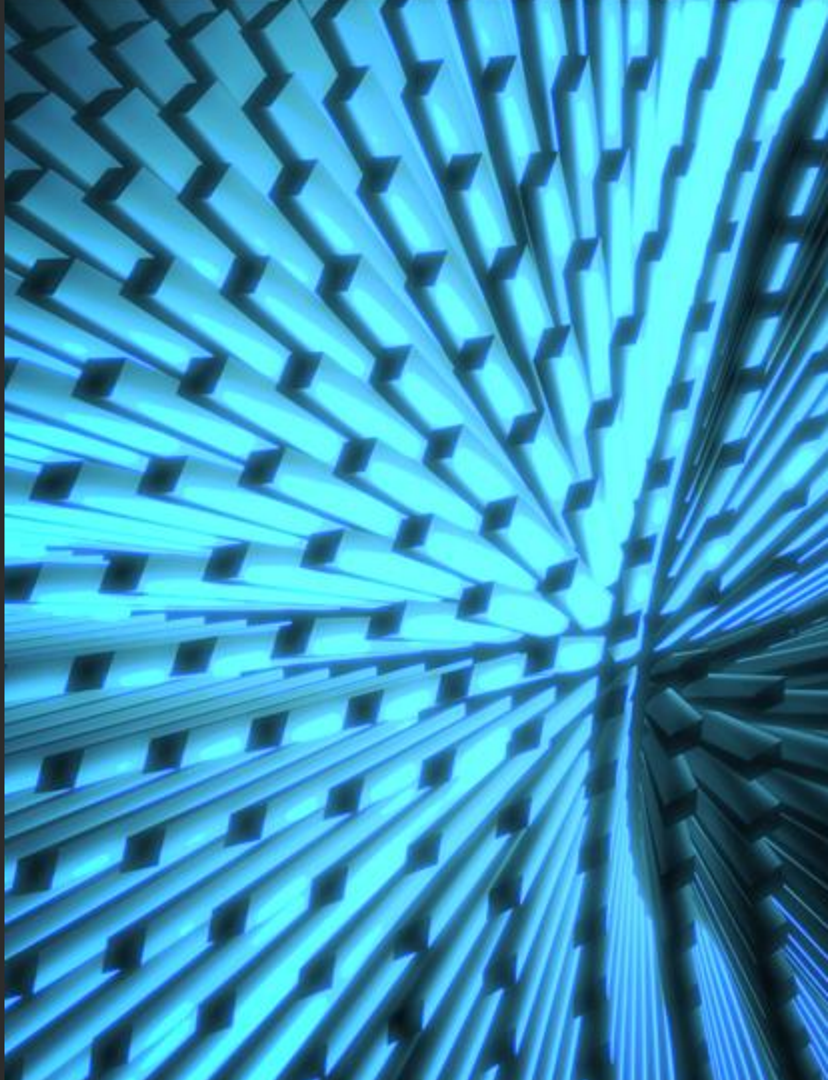
Giovanni Amorin (PUC-Rio, Brazil)

Alexandre Street (PUC-Rio, Brazil)

Package: https://github.com/LAMPSPUC/ApplicationDrivenLearning.jl

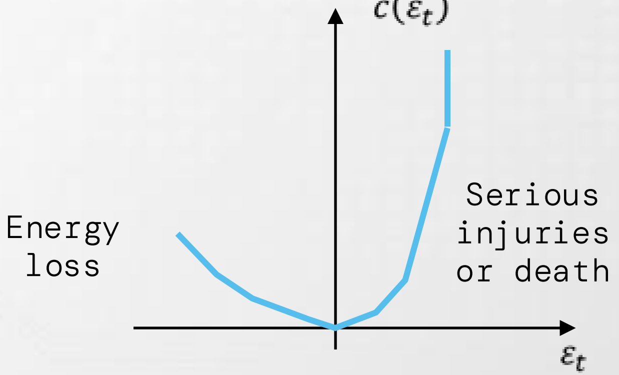
Theory paper: 10.1287/opre.2023.0565 (or https://arxiv.org/abs/2102.13273)

November 17th, 2025 - JuMP-dev 2025, Auckland, New Zealand



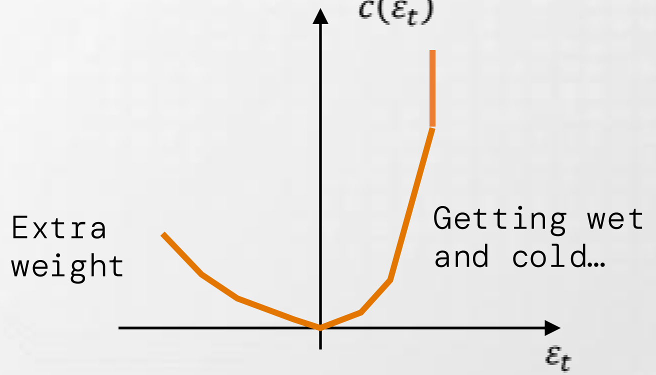
Motivation: Asymmetric costs

- **Parkour:** forecasted targets are biased

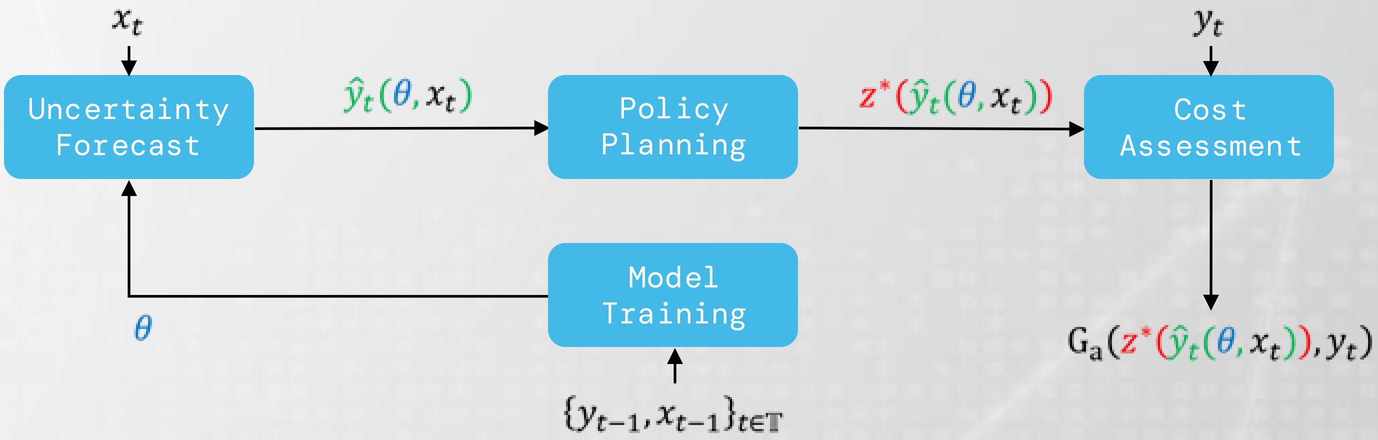


Motivation: Asymmetric costs

- **Weather:** forecasted targets are biased

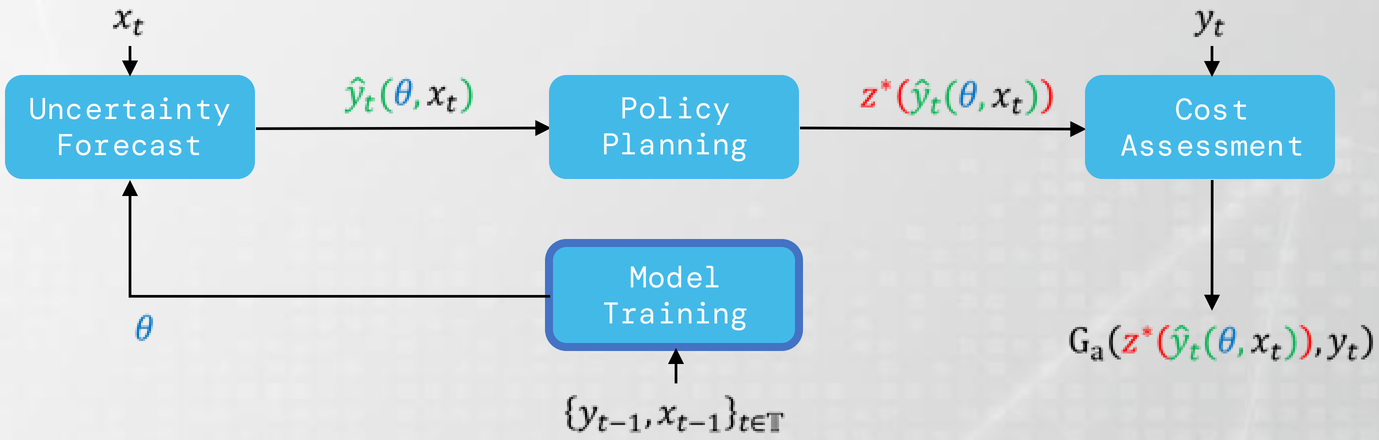


The general model: Open-Loop



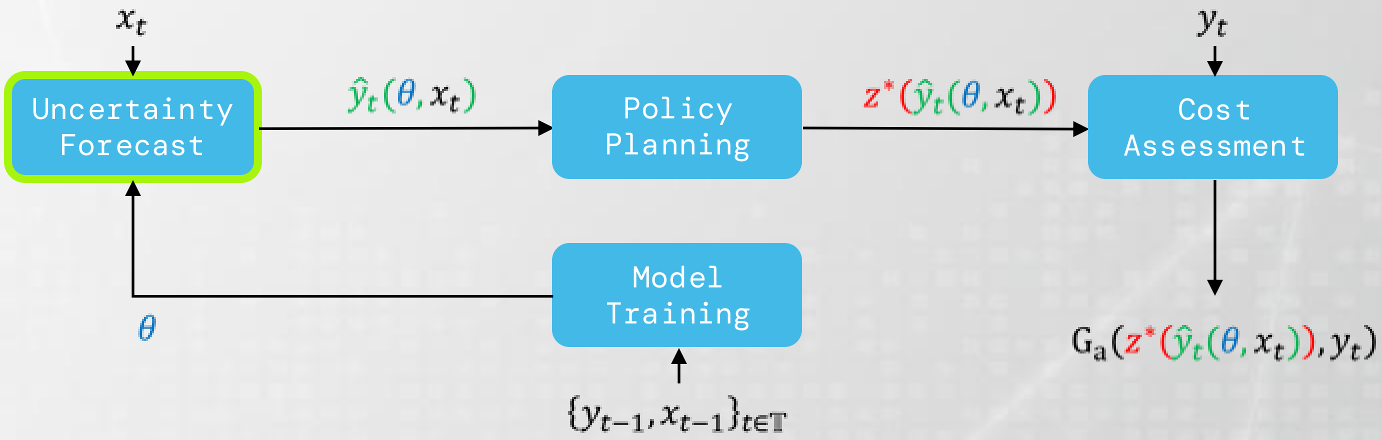
Train Load Forecast and Reserve Model
Forecast Loads and Reserve Requirements
Plan the Operation
Re-Dispatch Resources in Real Time

The general model: Open-Loop



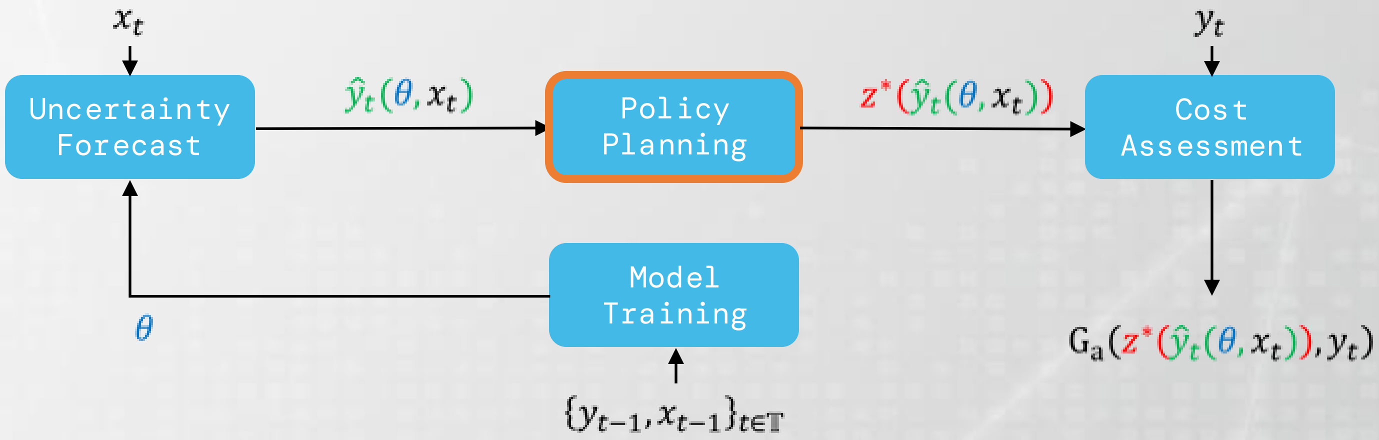
- Train Load Forecast and Reserve Model
- Forecast Loads and Reserve Requirements
- Plan the Operation
- Re-Dispatch Resources in Real Time

The general model: Open-Loop



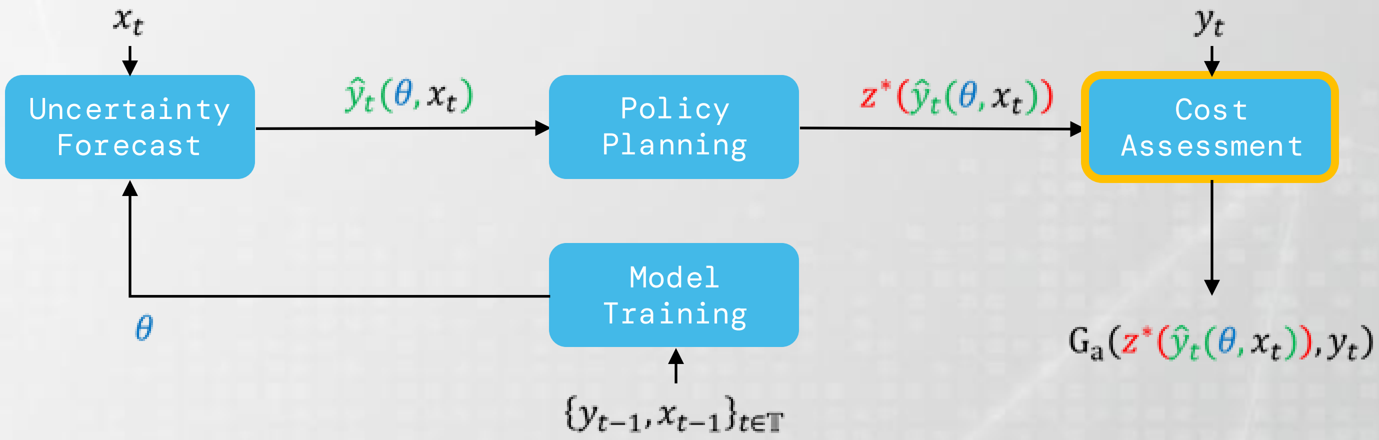
Train Load Forecast and Reserve Model
Forecast Loads and Reserve Requirements
Plan the Operation
Re-Dispatch Resources in Real Time

The general model: Open-Loop



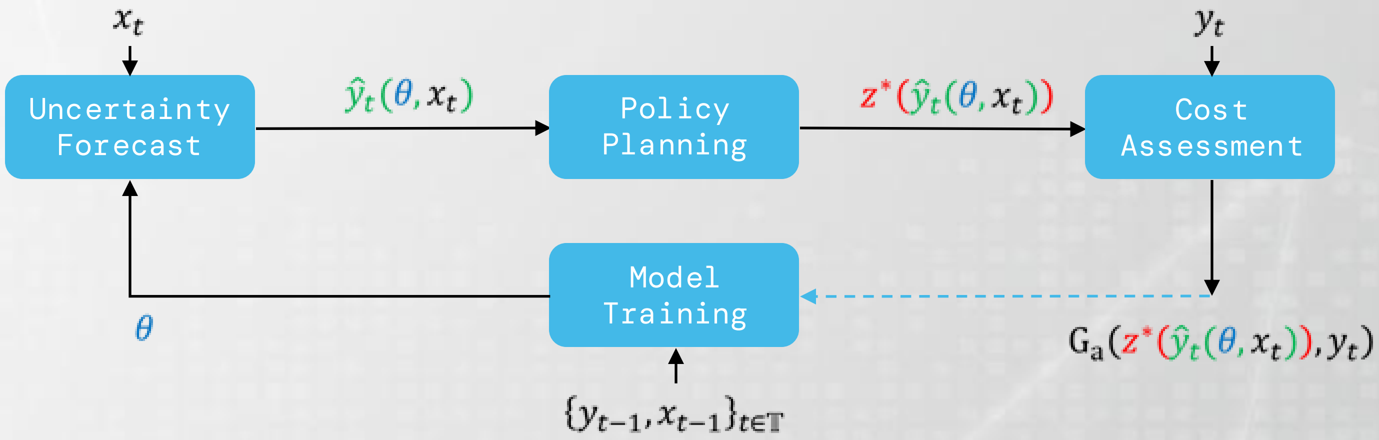
Train Load Forecast and Reserve Model
Forecast Loads and Reserve Requirements
Plan the Operation
Re-Dispatch Resources in Real Time

The general model: Open-Loop



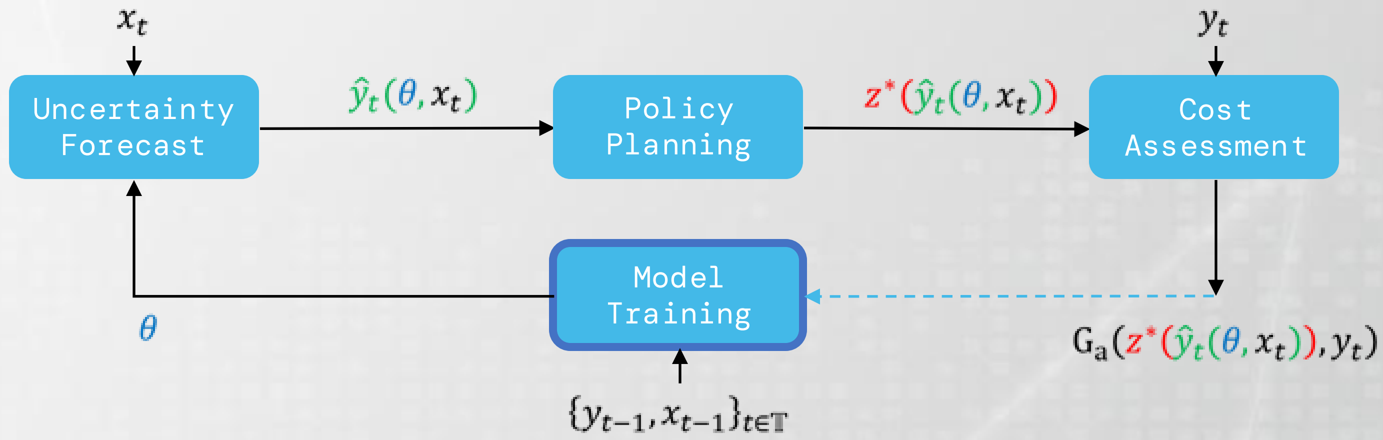
Train Load Forecast and Reserve Model
Forecast Loads and Reserve Requirements
Plan the Operation
Re-Dispatch Resources in Real Time

The general model: Closing the Loop



Train Load Forecast and Reserve Model
Forecast Loads and Reserve Requirements
Plan the Operation
Re-Dispatch Resources in Real Time

The general model: Closing the Loop



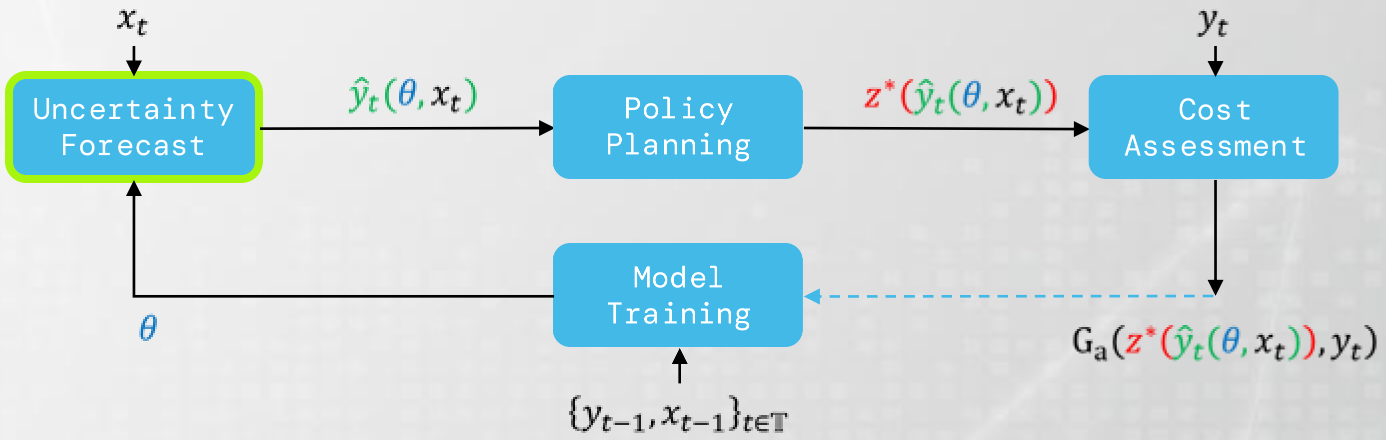
- Train Load Forecast and Reserve Model
- Forecast Loads and Reserve Requirements
- Plan the Operation
- Re-Dispatch Resources in Real Time

$$\boxed{\theta_T \in \arg\min_{\theta\in\Theta,\hat{y}_t,z_t^*}} \frac{1}{T}\sum_{t\in\mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \arg\min_{z\in Z} G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$

The general model: Closing the Loop



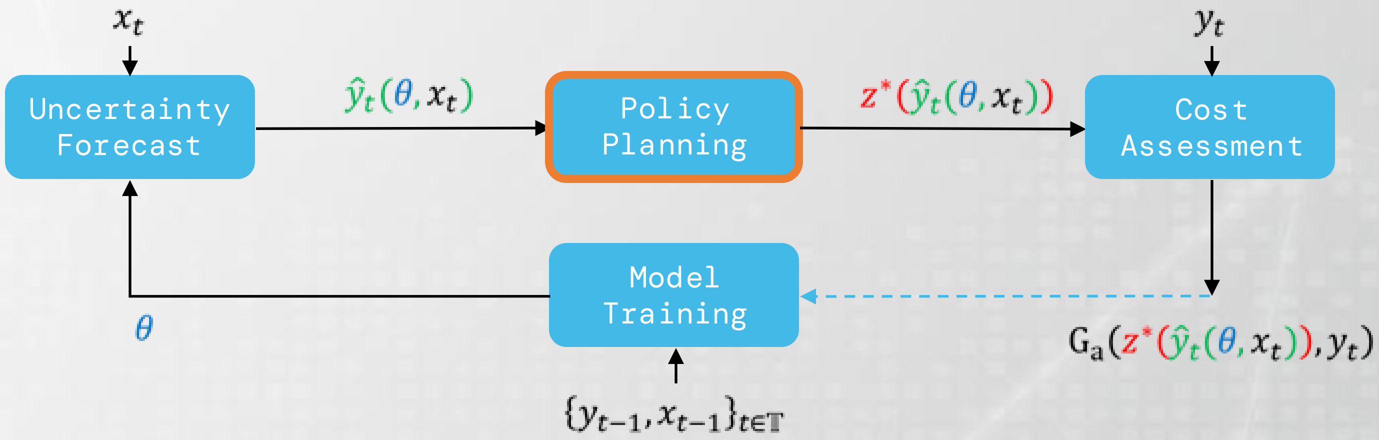
- Train Load Forecast and Reserve Model
- Forecast Loads and Reserve Requirements
- Plan the Operation
- Re-Dispatch Resources in Real Time

$$\theta_T \in \underset{\theta\in\Theta,\hat{y}_t,z_t^*}{\arg\min} \quad \frac{1}{T}\sum_{t\in\mathbb{T}} G_a(z_t^*, y_t)$$

$$\boxed{s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}}$$

$$z_t^* \in \underset{z\in Z}{\arg\min}\, G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$

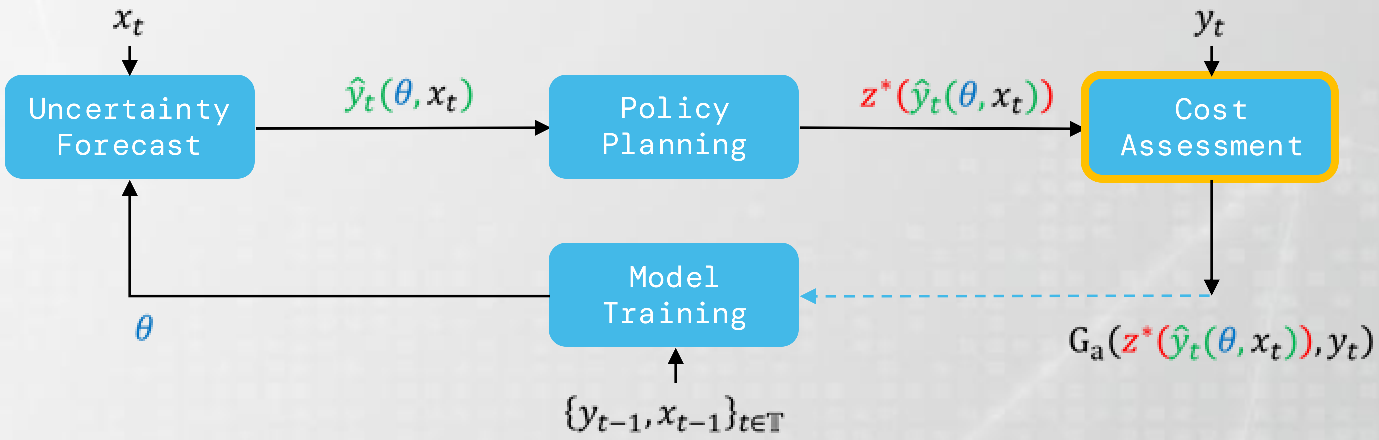
The general model: Closing the Loop



- Train Load Forecast and Reserve Model
- Forecast Loads and Reserve Requirements
- Plan the Operation
- Re-Dispatch Resources in Real Time

$$
\begin{aligned}
\theta_T \in \underset{\theta \in \Theta, \hat{y}_t, z_t^*}{\arg\min} \quad & \frac{1}{T} \sum_{t \in \mathbb{T}} G_a(z_t^*, y_t) \\
s.t. \quad & \hat{y}_t = \Psi(\theta, x_t) \; \forall t \in \mathbb{T} \\
& \boxed{z_t^* \in \underset{z \in Z}{\arg\min}\, G_p(z, \hat{y}_t) \; \forall t \in \mathbb{T}}
\end{aligned}
$$

The general model:
Closing the Loop



Train Load Forecast and Reserve Model
Forecast Loads and Reserve Requirements
Plan the Operation
Re-Dispatch Resources in Real Time

$$
\begin{aligned}
\theta_T \in \arg\min_{\theta\in\Theta,\hat{y}_t,z_t^*} \quad & \boxed{\frac{1}{T}\sum_{t\in\mathbb{T}} G_a(z_t^*, y_t)} \\
s.t. \quad & \hat{y}_t = \Psi(\theta, x_t)\ \forall t \in \mathbb{T} \\
& z_t^* \in \arg\min_{z\in Z} G_p(z, \hat{y}_t)\ \forall t \in \mathbb{T}
\end{aligned}
$$

Convergence

Assumptions

- Policy Planning has a unique solution.
- The set Z is non-empty and bounded
- Feasible set of the dual of Cost Assessment function is non-empty and bounded
- Forecasting function is continuous in both arguments
- Forecast parameter are in Θ, which is compact
- Forecast target Y_t which is stationary, ergodic and integrable
- Context X_t is a measurable function of Y_t

Define $\zeta(y) := \operatorname{argmin}_{z \in Z} G_p(z, y)$

Result

Hence $\lim_{T \to \infty} d(\theta_T, S^*) = 0$ wp1, for $S^* = \operatorname{argmin}_{\theta \in \Theta} \mathbb{E}\left[G_a\big(\zeta(\Psi(\theta, X)), Y\big)\right]$

For more

Home > Operations Research > Vol. 73, No. 1 >

Application-Driven Learning: A Closed-Loop Prediction and Optimization Approach Applied to Dynamic Reserves and Demand Forecasting

Joaquim Dias Garcia, Alexandre Street, Tito Homem-de-Mello, Francisco D. Muñoz

Published Online: 9 Sep 2024 | https://doi.org/10.1287/opre.2023.0565

Solution method: MILP

$$\theta_T \in \arg\min_{\theta\in\Theta,\hat{y}_t,z_t^*} \frac{1}{T}\sum_{t\in\mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \arg\min_{z\in Z} G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$

+

Linear models

$$G_i(z,y) = c_i^\top z + Q_i(z,y)$$

$$Q_i(z,y) = \min_u \{q_i^\top u \mid W_i u \geq b_i - H_i z + F_i y\}$$

+

KKT based MPEC reformulation

=

$$\min_{\theta\in\Theta,\hat{y}_t,z_t^*,u_t,\pi_t} \frac{1}{T}\sum_{t\in\mathbb{T}} [c_a^\top z_t^* + Q_a(z_t^*, y_t)]$$

$$s.t. \quad \forall t \in \mathbb{T}:$$

$$\hat{y}_t = \Psi(\theta, x_t)$$

$$W_p y_t + H_p z_t^* \geq b_p + F_p \hat{y}_t$$

$$A z_t^* \geq h$$

$$W_p{}^\top \pi_t = q_p$$

$$H_p^\top \pi_t + A^\top \mu_t = c_p$$

$$\pi_t, \mu_t \geq 0$$

$$\pi_t \perp W_p u_t + H_p z_t^* - b_p - F_p \hat{y}_t$$

$$\mu_t \perp A z_t^* - h$$

Solution method: MILP with BilevelJuMP.jl

$$
\begin{aligned}
\min_{\theta,\hat{y}_t,z_t,u_t^a,u_t} \quad & \frac{1}{T}\sum_{t\in\mathbb{T}} c_a^\top z_t^* + q_a^\top u_t^a \\
s.t. \quad & \forall t \in \mathbb{T}: \\
& C\theta \geq g \\
& \hat{y}_t = \theta^\top x_t \\
& W_a u_t^a \geq b_a - H_a z_t + F_a y_t \\
& z_t \in \arg\min_{z_t,u_t} c_p^\top z_t + q_p^\top u_t \\
& \qquad W_p u_t \geq b_p - H_p z_t + F_p \hat{y}_t \\
& \qquad A z_t \geq h
\end{aligned}
$$

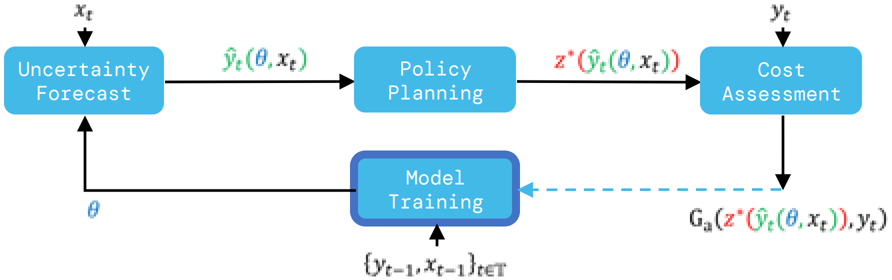
```julia
m = BilevelModel()

@variable(Upper(m), θ[1:L])
@variable(Upper(m), y[1:M,1:T])
@variable(Lower(m), u[1:N,1:T])
@variable(Lower(m), z[1:P,1:T])
@variable(Upper(m), ua[1:Q,1:T])

@objective(Upper(m),
    Min, 1/T * ∑(ca'z[:,t] + qa'ua[:,t] for t ∈ T))
@constraint(Upper(m), C*θ ≥ g)
for t ∈ T
@constraint(Upper(m),
    y_hat[:,t] = θ'x[:,t])
@constraint(Upper(m),
    Wa * ua[:,t] ≥ ba - Ha * z[:,t] + Fa * y[:,t])
end

@objective(Lower(m),
    Min, ∑(cp'z[:,t] + qp'u[:,t] for t ∈ T))
for t ∈ T
@constraint(Lower(m),
    A * z[:,t] ≥ h)
@constraint(Lower(m),
    Wp * u[:,t] ≥ bp - Hp * z[:,t] + Fp * y_hat[:,t])
end
```

Solution method:
Scalable Heuristic

$$\boxed{\theta_T \in \underset{\theta \in \Theta, \hat{y}_t, z_t^*}{\arg\min}} \frac{1}{T} \sum_{t \in \mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$z_t^* \in \underset{z \in Z}{\arg\min}\, G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}$$



Algorithm 1: Pseudo algorithm

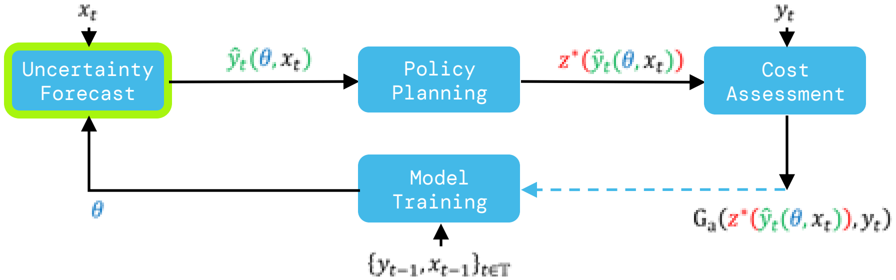
Result: Optimized θ
Initialize θ ;
while *Not converged* **do**
  $\boxed{\text{Update } \theta;}$
  for $t \in \mathbb{T}$ **do**
    Forecast: $\hat{y}_t \leftarrow \Psi(\theta, x_t)$;
    Plan Policy: $z_t^* \leftarrow \arg\min_{z \in Z} G_p(z, \hat{y}_t)$;
    Cost Assessment: $cost_t \leftarrow G_a(z_t^*, y_t)$
  end
  Compute cost: $cost(\theta) \leftarrow \sum_{t \in \mathbb{T}}(cost_t)$
end

Solution method:
Scalable Heuristic

$$\theta_T \in \underset{\theta\in\Theta,\hat{y}_t,z_t^*}{\arg\min} \quad \frac{1}{T}\sum_{t\in\mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \boxed{\hat{y}_t = \Psi(\theta, x_t)\ \forall t \in \mathbb{T}}$$

$$z_t^* \in \underset{z\in Z}{\arg\min}\, G_p(z, \hat{y}_t)\ \forall t \in \mathbb{T}$$



Algorithm 1: Pseudo algorithm

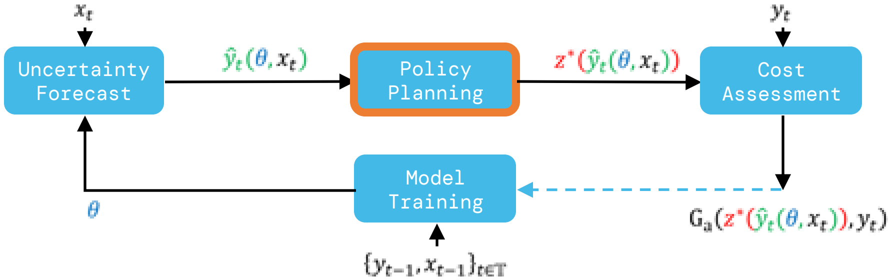
Result: Optimized θ
Initialize θ ;
while *Not converged* **do**
  Update θ;
  for $t \in \mathbb{T}$ **do**
    $\boxed{\text{Forecast: } \hat{y}_t \leftarrow \Psi(\theta, x_t);}$
    Plan Policy: $z_t^* \leftarrow \arg\min_{z\in Z} G_p(z, \hat{y}_t)$;
    Cost Assessment: $cost_t \leftarrow G_a(z_t^*, y_t)$
  end
  Compute cost: $cost(\theta) \leftarrow \sum_{t\in\mathbb{T}}(cost_t)$
end

Solution method:
Scalable Heuristic

$$\theta_T \in \underset{\theta\in\Theta,\hat{y}_t,z_t^*}{\arg\min} \quad \frac{1}{T}\sum_{t\in\mathbb{T}} G_a(z_t^*, y_t)$$

$$s.t. \quad \hat{y}_t = \Psi(\theta, x_t) \ \forall t \in \mathbb{T}$$

$$\boxed{z_t^* \in \underset{z\in Z}{\arg\min}\, G_p(z, \hat{y}_t) \ \forall t \in \mathbb{T}}$$



Algorithm 1: Pseudo algorithm

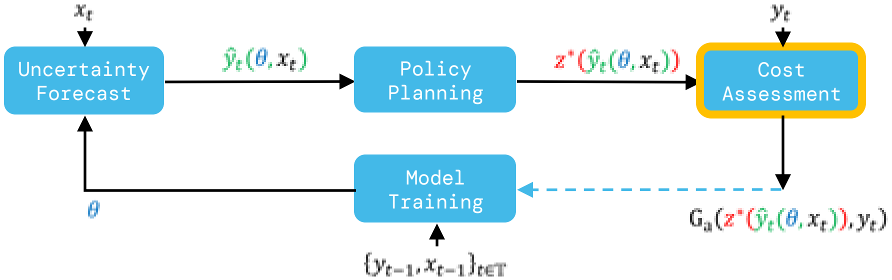
Result: Optimized θ
Initialize θ ;
while *Not converged* **do**
  Update θ;
  for $t \in \mathbb{T}$ **do**
    Forecast: $\hat{y}_t \leftarrow \Psi(\theta, x_t)$;
    $\boxed{\text{Plan Policy: } z_t^* \leftarrow \arg\min_{z\in Z} G_p(z, \hat{y}_t);}$
    Cost Assessment: $cost_t \leftarrow G_a(z_t^*, y_t)$
  end
  Compute cost: $cost(\theta) \leftarrow \sum_{t\in\mathbb{T}}(cost_t)$
end

Solution method:
Scalable Heuristic

$$\theta_T \in \arg\min_{\theta\in\Theta,\hat{y}_t,z_t^*} \boxed{\frac{1}{T}\sum_{t\in\mathbb{T}} G_a(z_t^*, y_t)}$$

$$\text{s.t.} \quad \hat{y}_t = \Psi(\theta, x_t) \;\; \forall t \in \mathbb{T}$$

$$z_t^* \in \arg\min_{z\in Z} G_p(z, \hat{y}_t) \;\; \forall t \in \mathbb{T}$$



Algorithm 1: Pseudo algorithm

```
Result: Optimized θ
Initialize θ ;
while Not converged do
    Update θ;
    for t ∈ T do
        Forecast: ŷ_t ← Ψ(θ, x_t);
        Plan Policy: z*_t ← arg min_{z∈Z} G_p(z, ŷ_t);
        Cost Assessment: cost_t ← G_a(z*_t, y_t)
    end
    Compute cost: cost(θ) ← Σ_{t∈T}(cost_t)
end
```

Solution method: Scalable Heuristic

Naïve choice of "Update θ" : zero order methods like Nelder Mead

Pros:

- Simplicity: forecast, planning and assess models can be basically anything as long as we can compute final costs efficiently.

Cons:

- Search method for cost minimization not smartly guided by problem structure and can take long time.
- For high-cardinality model parameters, can become intractable.

Solution method:
Scalable Heuristic

Another choice of "Update θ" : first order methods like Gradient Descent

$$C_t = Cost\big(z^*(\Psi(\theta, x_t)), y_t\big)$$

$$\hat{y}_t = \Psi(\theta, x_t)$$

$$\frac{\partial C_t}{\partial \theta} = \frac{\partial C_t}{\partial z^*} \cdot \frac{\partial z^*}{\partial \hat{y}_t} \cdot \frac{\partial \Psi(\theta, x_t)}{\partial \theta}$$

Solution from the dual problem

JUMP

Differentiable Optimization

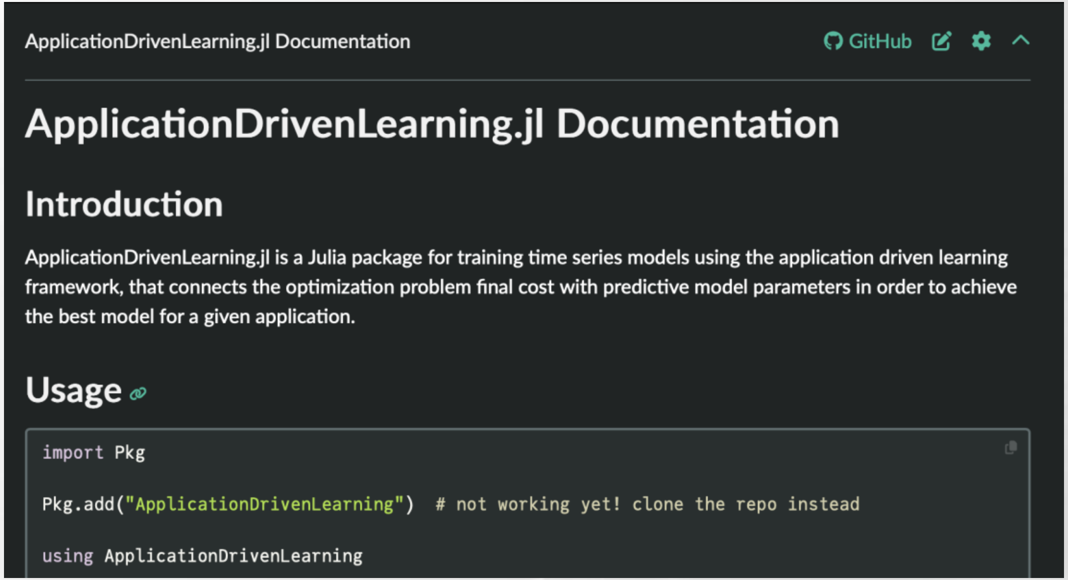
DiffOpt.jl

Automatic Differentiation

flux

The New Package

- The ***ApplicationDrivenLearning.jl*** package presents an easy way of training models in the closed loop fashion.



ApplicationDrivenLearning.jl Documentation GitHub

ApplicationDrivenLearning.jl Documentation

Introduction

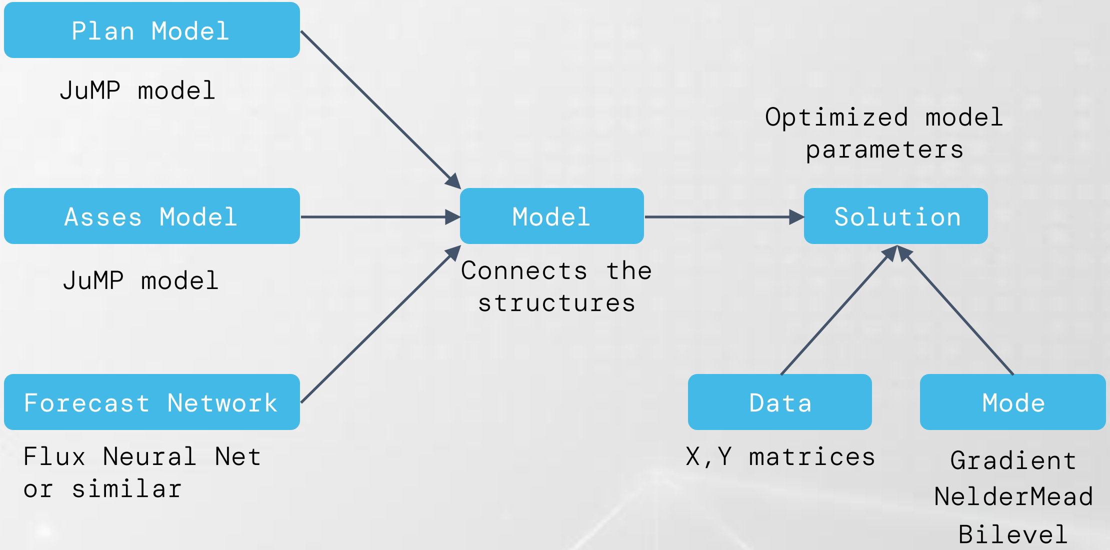
ApplicationDrivenLearning.jl is a Julia package for training time series models using the application driven learning framework, that connects the optimization problem final cost with predictive model parameters in order to achieve the best model for a given application.

Usage

```julia
import Pkg

Pkg.add("ApplicationDrivenLearning")  # not working yet! clone the repo instead

using ApplicationDrivenLearning
```

The Package

- The ***ApplicationDrivenLearning.jl*** package presents an easy way of training models in the closed loop fashion.



The Package

- **Solution methods**
  - Bilevel Optimization: JuMP.jl + BilevelJuMP.jl
  - Nelder Mead: JuMP.jl + Optim.jl
  - Gradient Descent: JuMP.jl + DiffOpt.jl + Flux.jl
- **Problem classes**
  - Linear & Quadratic
  - Conic
  - Non-Linear
- **Other features**
  - MPI parallelism
  - Solver agnostic (use your favorite)

The Package: Example

```julia
# data
X = ones(30, 1)
Y = rand(DiscreteUniform(0, 1), (30, 1)) .* 2

# main model and policy / forecast variables
model = ApplicationDrivenLearning.AppDrivenModel()
@variables(model, begin
    z, ApplicationDrivenLearning.Policy
    θ, ApplicationDrivenLearning.Forecast
end)

# plan model
@variables(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 0
    c2 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 100 * (θ.plan-z.plan)
    c2 ≥ 20 * (z.plan-θ.plan)
end)
@objective(ApplicationDrivenLearning.Plan(model), Min, 10*z.plan + c1 + c2)

# assess model
@variables(ApplicationDrivenLearning.Assess(model), begin
    c1 ≥ 0
    c2 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Assess(model), begin
    c1 ≥ 100 * (θ.assess-z.assess)
    c2 ≥ 20 * (z.assess-θ.assess)
end)
@objective(ApplicationDrivenLearning.Assess(model), Min, 10*z.assess + c1 + c2)
```

The Package: Example

```julia
# data
X = ones(30, 1)
Y = rand(DiscreteUniform(0, 1), (30, 1)) .* 2

# main model and policy / forecast variables
model = ApplicationDrivenLearning.AppDrivenModel()
@variables(model, begin
    z, ApplicationDrivenLearning.Policy
    θ, ApplicationDrivenLearning.Forecast
end)

# plan model
@variables(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 0
    c2 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 100 * (θ.plan-z.plan)
    c2 ≥ 20 * (z.plan-θ.plan)
end)
@objective(ApplicationDrivenLearning.Plan(model), Min, 10*z.plan + c1 + c2)

# assess model
@variables(ApplicationDrivenLearning.Assess(model), begin
    c1 ≥ 0
    c2 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Assess(model), begin
    c1 ≥ 100 * (θ.assess-z.assess)
    c2 ≥ 20 * (z.assess-θ.assess)
end)
@objective(ApplicationDrivenLearning.Assess(model), Min, 10*z.assess + c1 + c2)
```

The Package: Example

```julia
# data
X = ones(30, 1)
Y = rand(DiscreteUniform(0, 1), (30, 1)) .* 2

# main model and policy / forecast variables
model = ApplicationDrivenLearning.AppDrivenModel()
@variables(model, begin
    z, ApplicationDrivenLearning.Policy
    θ, ApplicationDrivenLearning.Forecast
end)

# plan model
@variables(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 0
    c2 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 100 * (θ.plan-z.plan)
    c2 ≥ 20 * (z.plan-θ.plan)
end)
@objective(ApplicationDrivenLearning.Plan(model), Min, 10*z.plan + c1 + c2)

# assess model
@variables(ApplicationDrivenLearning.Assess(model), begin
    c1 ≥ 0
    c2 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Assess(model), begin
    c1 ≥ 100 * (θ.assess-z.assess)
    c2 ≥ 20 * (z.assess-θ.assess)
end)
@objective(ApplicationDrivenLearning.Assess(model), Min, 10*z.assess + c1 + c2)
```

The Package: Example

```julia
# data
X = ones(30, 1)
Y = rand(DiscreteUniform(0, 1), (30, 1)) .* 2

# main model and policy / forecast variables
model = ApplicationDrivenLearning.AppDrivenModel()
@variables(model, begin
    z, ApplicationDrivenLearning.Policy
    θ, ApplicationDrivenLearning.Forecast
end)

# plan model
@variables(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 0
    c2 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Plan(model), begin
    c1 ≥ 100 * (θ.plan-z.plan)
    c2 ≥ 20 * (z.plan-θ.plan)
end)
@objective(ApplicationDrivenLearning.Plan(model), Min, 10*z.plan + c1 + c2)

# assess model
@variables(ApplicationDrivenLearning.Assess(model), begin
    c1 ≥ 0
    c2 ≥ 0
end)
@constraints(ApplicationDrivenLearning.Assess(model), begin
    c1 ≥ 100 * (θ.assess-z.assess)
    c2 ≥ 20 * (z.assess-θ.assess)
end)
@objective(ApplicationDrivenLearning.Assess(model), Min, 10*z.assess + c1 + c2)
```

The Package: Example

```julia
# basic setting
set_optimizer(model, HiGHS.Optimizer)
set_silent(model)

# forecast model
nn = Chain(Dense(1 => 1; bias=false))
ApplicationDrivenLearning.set_forecast_model(model, nn)

# training and getting solution
sol = ApplicationDrivenLearning.train!(
    model,
    X,
    Y,
    ApplicationDrivenLearning.Options(
        ApplicationDrivenLearning.NelderMeadMode
    )
)
sol.params
sol.cost / 30 | 38.666676f0
```

The Package: Example

```julia
# basic setting
set_optimizer(model, HiGHS.Optimizer)
set_silent(model)

# forecast model
nn = Chain(Dense(1 => 1; bias=false))
ApplicationDrivenLearning.set_forecast_model(model, nn)

# training and getting solution
sol = ApplicationDrivenLearning.train!(
    model,
    X,
    Y,
    ApplicationDrivenLearning.Options(
        ApplicationDrivenLearning.NelderMeadMode
    )
)
sol.params
sol.cost / 30 | 38.666676f0
```

The Package: Example

```julia
# basic setting
set_optimizer(model, HiGHS.Optimizer)
set_silent(model)

# forecast model
nn = Chain(Dense(1 => 1; bias=false))
ApplicationDrivenLearning.set_forecast_model(model, nn)

# training and getting solution
sol = ApplicationDrivenLearning.train!(
    model,
    X,
    Y,
    ApplicationDrivenLearning.Options(
        ApplicationDrivenLearning.NelderMeadMode
    )
)
sol.params
sol.cost / 30 | 38.666676f0
```

Application Driven Model for Load and Reserve

Train Model

- Real Time Cost Assessment
Generation, Reserve, Shed, Spill

Forecast Demand and Reserve

- Load Balance
- Network Flows
- Generation redispatch
- Bounds

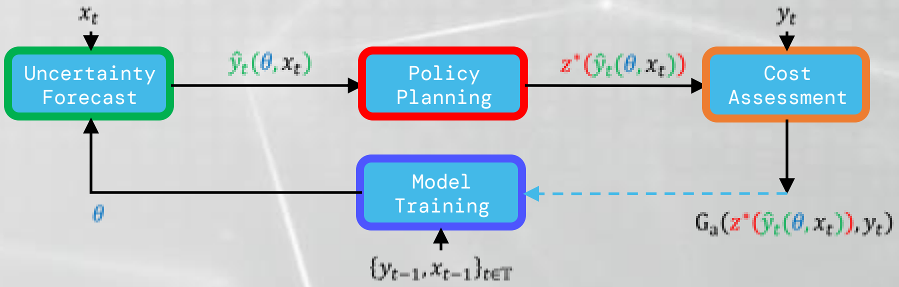
- Policy Planning Cost
Generation, Reserve, Shed, Spill

s.t.
- Load Balance
- Network Flows
- Zonal Reserve Requirement
- Reserve and Generation Allocation
- Bounds

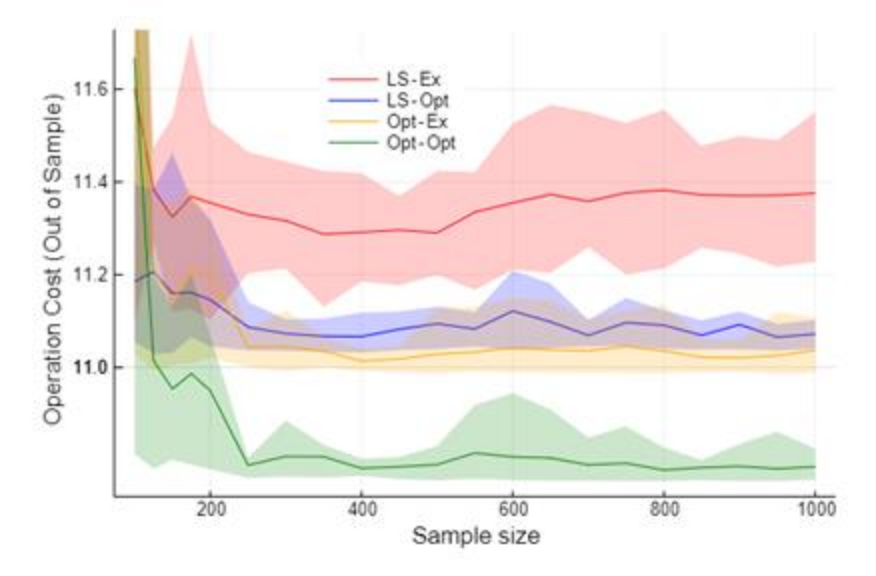
$$\boxed{\theta_T \in \underset{\theta\in\Theta,\hat{y}_t,z_t^*}{\arg\min}} \boxed{\frac{1}{T}\sum_{t\in\mathbb{T}} G_a(z_t^*, y_t)}$$

$$\text{s.t.} \quad \boxed{\hat{y}_t = \Psi(\theta, x_t)\ \forall t \in \mathbb{T}}$$

$$\boxed{z_t^* \in \underset{z\in Z}{\arg\min}\, G_p(z, \hat{y}_t)\ \forall t \in \mathbb{T}}$$

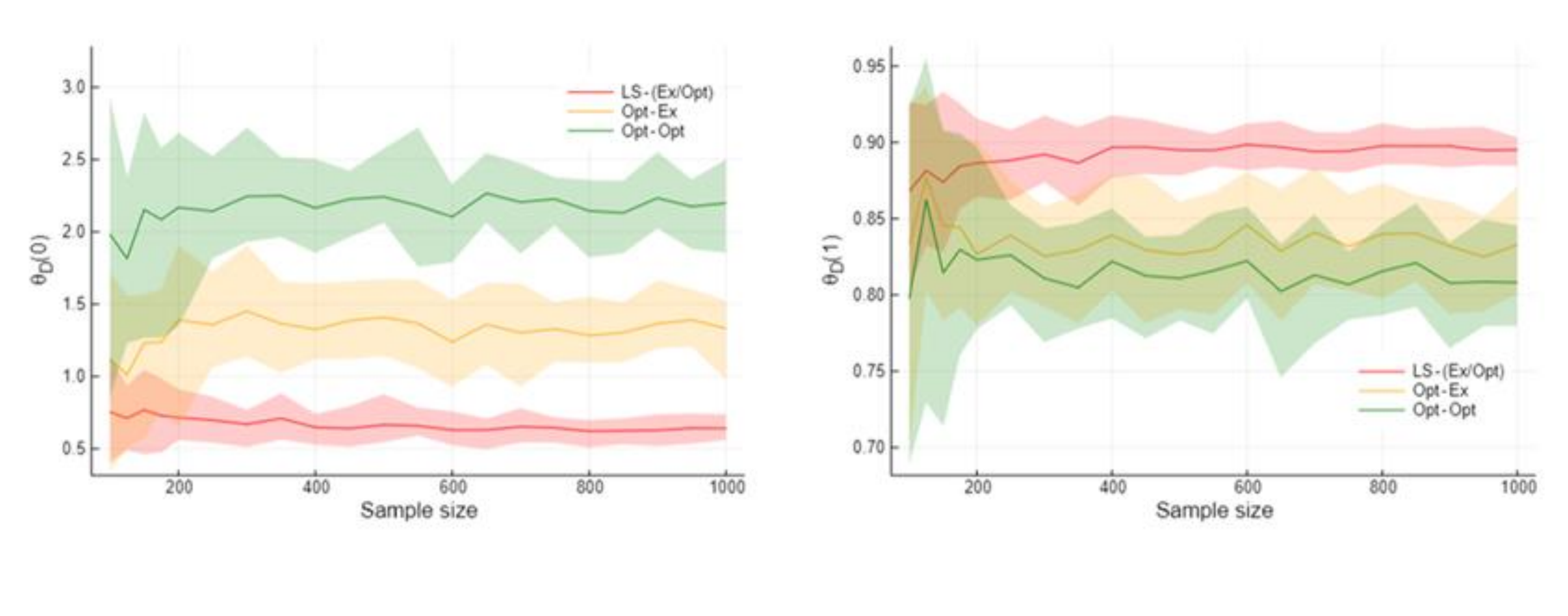


Case Study:
Convergence of Objective

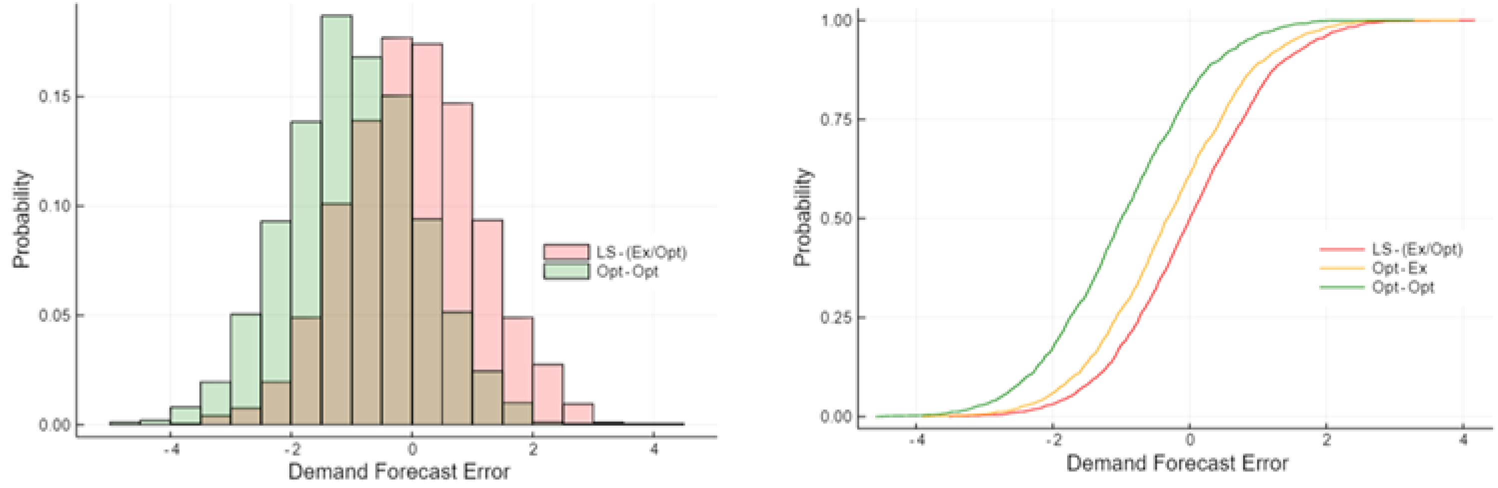


- LS-Ex (red)
  Least-Squares load
  Exogenous reserves
- LS-Opt (blue)
  Least-Squares load
  Optimized reserves
- Opt-Ex (yellow)
  Optimized load
  Exogenous reserves
- Opt-Opt (green)
  Optimized load
  Optimized reserves

Case Study:
Convergence of Solution



Case Study:
Biased forecast



Case Study: Multiple methods

N Buses	LS	MPI-NM	MPI-GD	GD
24	7.608,62	7.617,16 (-0,11%)	7.627,59 (-0,25%)	**7.604,44 (0,05%)**
118	11.515,94	11.177,89 (2,94%)	**10.572,90 (8,19%)**	10.579,62 (8,13%)
179	123.313,02	113.038,81 (8,33%)	**82.549,92 (33,06%)**	90.703,52 (26,44%)
240	801.135,26	-	**760.829,71 (5,03%)**	767.958,60 (4,14%)
300	82.968,84	-	76.625,42 (7,65%)	**76.313,26 (8,02%)**
500	29.154,24	-	**29.051,50 (0,35%)**	29.061,10 (0,32%)
588	33.496,56	-	**29.362,07 (12,34%)**	30.577,54 (8,71%)
793	48.403,34	-	**37.940,31 (21,62%)**	42.558,32 (12,08%)
1354	176.562,56	-	**172.565,10 (2,26%)**	-

The end

- **Package:**
  - github.com/LAMPSPUC/ApplicationDrivenLearning.jl
  - Friendly interface
  - Multiple solution methods and solvers
  - HPC Ready
- **Method:**
  - Outperformed open-loop framework
  - Optimal forecasts are BIASED
  - Meaningful improvements even in very large-scale systems
  - Can be used in practice