KTH ROYAL INSTITUTE
OF TECHNOLOGY

Distributed Stochastic Programming

with Applications to Large-Scale Hydropower Operations

Martin Biel

KTH - Royal Institute of Technology

Licentiate thesis, November 29, 2019

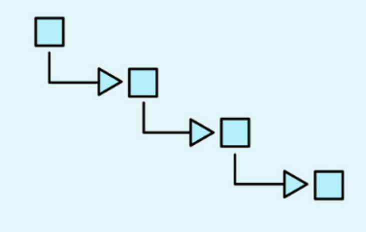
Motivation - Hydropower operations

Motivation - Hydropower operations

- Hydroelectric power production

Motivation - Hydropower operations

- Hydroelectric power production
- Spatial dependence

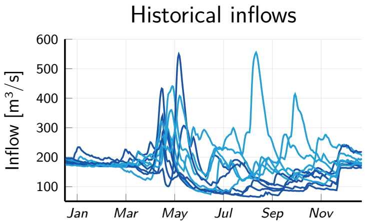


Motivation - Hydropower operations

- Hydroelectric power production
- Spatial dependence
- Temporal dependence

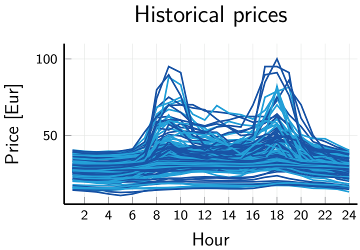
Motivation - Hydropower operations

- Uncertain local inflow



Motivation - Hydropower operations

- Uncertain local inflow
- Uncertain electricity price



Motivation - Hydropower operations

- Uncertain local inflow
- Uncertain electricity price
- Uncertain renewable production

Motivation - Hydropower operations

- Store energy in water reservoirs

Motivation - Optimization models

- Decision support: formulate and solve optimization models

Motivation - Optimization models

- Decision support: formulate and solve optimization models
- Common: trade-off between accuracy and computation time

Motivation - Optimization models

- Decision support: formulate and solve optimization models
- Common: trade-off between accuracy and computation time
- Aim: **provide reliable decision-support in a short amount of time**

Motivation - Optimization models

- Decision support: formulate and solve optimization models
- Common: trade-off between accuracy and computation time
- Aim: **provide reliable decision-support in a short amount of time**
  - Accurate models: optimal model reductions
  - Fast computations: scalable algorithms on commodity hardware

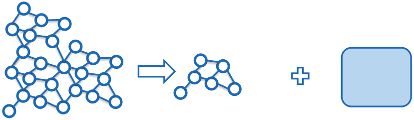


Figure: Manageable models

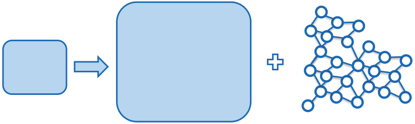


Figure: Scalable algorithms

Motivation - Stochastic programming

Mathematical framework for decision problems subjected to uncertainty

Motivation - Stochastic programming

Mathematical framework for decision problems subjected to uncertainty

Decision

Actions

- Investments
- Schedules
- Orders

Motivation - Stochastic programming

Mathematical framework for decision problems subjected to uncertainty

Decision $\longrightarrow$ Observation

Actions	**Uncertainties**
• Investments	• Demand
• Schedules	• Weather conditions
• Orders	• Market price

Motivation - Stochastic programming

Mathematical framework for decision problems subjected to uncertainty

Decision ⟶ **Observation** ⟶ **Recourse**

Actions

- Investments
- Schedules
- Orders

Uncertainties

- Demand
- Weather conditions
- Market price

Actions

- Restock
- Reschedule
- Settle imbalances

Motivation - Stochastic programming

Stochastic programming for hydropower operations

- Order strategies in deregulated electricity markets
- Capacity expansion
- Coordination with renewable production
- Maintenance scheduling
- Seasonal planning: reservoir contents before spring flood

Contribution

- `StochasticPrograms.jl`: framework for stochastic programming

Contribution

- `StochasticPrograms.jl`: framework for stochastic programming
- Distributed stochastic programming for large-scale models

Contribution

- `StochasticPrograms.jl`: framework for stochastic programming
- Distributed stochastic programming for large-scale models
- Efficient implementations of structure-exploiting algorithms

Contribution

- `StochasticPrograms.jl`: framework for stochastic programming
- Distributed stochastic programming for large-scale models
- Efficient implementations of structure-exploiting algorithms
- Algorithmic innovations and software patterns

Contribution

- `StochasticPrograms.jl`: framework for stochastic programming
- Distributed stochastic programming for large-scale models
- Efficient implementations of structure-exploiting algorithms
- Algorithmic innovations and software patterns
- Detailed consideration of a hydropower problem

Outline

Outline

Preliminaries - Stochastic program

- First stage decision: x

Preliminaries - Stochastic program

- First stage decision: x
- Recourse decision: y

Preliminaries - Stochastic program

- First stage decision: x
- Recourse decision: y
- Uncertainty: $\xi(\omega) : \Omega \to \mathbb{R}^N$ random variable on the set of events Ω

Preliminaries - Stochastic program

- First stage decision: x
- Recourse decision: y
- Uncertainty: $\xi(\omega) : \Omega \to \mathbb{R}^N$ random variable on the set of events Ω

First stage

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Preliminaries - Stochastic program

- First stage decision: x
- Recourse decision: y
- Uncertainty: $\xi(\omega) : \Omega \to \mathbb{R}^N$ random variable on the set of events Ω

First stage

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Preliminaries - Stochastic program

- First stage decision: x
- Recourse decision: y
- Uncertainty: $\xi(\omega) : \Omega \to \mathbb{R}^N$ random variable on the set of events Ω

First stage

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Second stage

$$\begin{aligned} Q(x, \xi(\omega)) = \min_{y \in \mathbb{R}^m} \quad & q_\omega^T y \\ \text{s.t.} \quad & W y = h_\omega - T_\omega x \\ & y \geq 0 \end{aligned}$$

Preliminaries - Stochastic program

- First stage decision: x
- Recourse decision: y
- Uncertainty: $\xi(\omega) : \Omega \to \mathbb{R}^N$ random variable on the set of events Ω

First stage

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Second stage

$$\begin{aligned} Q(x, \xi(\omega)) = \min_{y \in \mathbb{R}^m} \quad & q_\omega^T y \\ \text{s.t.} \quad & W y = h_\omega - T_\omega x \\ & y \geq 0 \end{aligned}$$

Preliminaries - Stochastic program

- First stage decision: x
- Recourse decision: y
- Uncertainty: $\xi(\omega) : \Omega \to \mathbb{R}^N$ random variable on the set of events Ω

First stage

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Second stage

$$\begin{aligned} Q(x, \xi(\omega)) = \min_{y\in\mathbb{R}^m} \quad & q_\omega^T y \\ \text{s.t.} \quad & W y = h_\omega - T_\omega x \\ & y \geq 0 \end{aligned}$$

Preliminaries - Stochastic program

- First stage decision: x
- Recourse decision: y
- Uncertainty: $\xi(\omega) : \Omega \to \mathbb{R}^N$ random variable on the set of events Ω

First stage

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Second stage

$$\begin{aligned} Q(x, \xi(\omega)) = \min_{y \in \mathbb{R}^m} \quad & q_\omega^T y \\ \text{s.t.} \quad & W y = h_\omega - T_\omega x \\ & y \geq 0 \end{aligned}$$

Preliminaries - Stochastic program

- First stage decision: x
- Recourse decision: y
- Uncertainty: $\xi(\omega) : \Omega \to \mathbb{R}^N$ random variable on the set of events Ω

First stage

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x + \mathbb{E}_\xi[Q(x, \xi(\omega))] \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Second stage

$$\begin{aligned} Q(x, \xi(\omega)) = \min_{y \in \mathbb{R}^m} \quad & q_\omega^T y \\ \text{s.t.} \quad & W y = h_\omega - T_\omega x \\ & y \geq 0 \end{aligned}$$

Preliminaries - Stochastic program

- First stage decision: x
- Recourse decision: y
- Uncertainty: $\xi(\omega) : \Omega \to \mathbb{R}^N$ random variable on the set of events Ω

First stage

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x + \mathbb{E}_\xi[Q(x, \xi(\omega))] \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Second stage

$$\begin{aligned} Q(x, \xi(\omega)) = \min_{y \in \mathbb{R}^m} \quad & q_\omega^T y \\ \text{s.t.} \quad & W y = h_\omega - T_\omega x \\ & y \geq 0 \end{aligned}$$

Preliminaries - Stochastic program

Definition (Linear two-stage stochastic program)

A *linear two-stage stochastic program* is given by

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x + \mathbb{E}_\xi[Q(x, \xi(\omega))] \\ \text{subject to} \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

where

$$\begin{aligned} Q(x, \xi(\omega)) = \min_{y \in \mathbb{R}^m} \quad & q_\omega^T y \\ \text{s.t.} \quad & T_\omega x + Wy = h_\omega \\ & y \geq 0. \end{aligned}$$

The optimal value is called the *value of the recourse problem* (VRP).

Preliminaries - Stochastic performance

Definition (Expected value decision)

Given

$$\bar{\xi} = \mathbb{E}_{\xi}[\xi(\omega)]$$

the *expected value decision* $\bar{x}$ associated with a given stochastic program is given by the solution to

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x + Q(x, \bar{\xi}) \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0. \end{aligned}$$

This problem is known as the *expected value problem*.

Preliminaries - Stochastic performance

Definition

The *expected result of the expected value decision*, or the EEV, is given by

$$EEV = c^T \bar{x} + \mathbb{E}_{\xi}[Q(\bar{x}, \xi(\omega))].$$

Preliminaries - Stochastic performance

Definition

The *expected result of the expected value decision*, or the EEV, is given by

$$EEV = c^T\bar{x} + \mathbb{E}_\xi[Q(\bar{x}, \xi(\omega))].$$

Definition

The *value of the stochastic solution*, or the VSS, is given by

$$VSS = VRP - EEV.$$

Preliminaries - Finite extensive form

- Ω finite

Preliminaries - Finite extensive form

- Ω finite
- ξ discrete random variable

Preliminaries - Finite extensive form

- Ω finite
- ξ discrete random variable

$$\begin{aligned} \underset{x\in\mathbb{R}^n, y_s\in\mathbb{R}^m}{\text{minimize}} \quad & c^T x + \sum_{s=1}^{n} \pi_s q_s^T y_s \\ \text{subject to} \quad & Ax = b \\ & T_s x + W y_s = h_s, \qquad s = 1, \dots, n \\ & x \geq 0, \; y_s \geq 0, \qquad s = 1, \dots, n \end{aligned}$$

Preliminaries - Finite extensive form

- Ω finite
- ξ discrete random variable

$$\begin{aligned} \underset{x\in\mathbb{R}^n, y_s\in\mathbb{R}^m}{\text{minimize}} \quad & c^T x + \sum_{s=1}^{n} \pi_s q_s^T y_s \\ \text{subject to} \quad & Ax = b \\ & T_s x + W y_s = h_s, \qquad s = 1, \dots, n \\ & x \geq 0, \; y_s \geq 0, \qquad s = 1, \dots, n \end{aligned}$$

Also commonly referred to as the *deterministic equivalent problem*, or the DEP.

Preliminaries - Sample average approximation

- Ω infinite

Preliminaries - Sample average approximation

- Ω infinite
- ξ continuous random variable

Preliminaries - Sample average approximation

- Ω infinite
- ξ continuous random variable
- Sample n scenarios $\omega_s,\ s = 1, \dots, n$ independently from Ω

Preliminaries - Sample average approximation

- Ω infinite
- ξ continuous random variable
- Sample n scenarios $\omega_s,\ s = 1,\dots,n$ independently from Ω

$$
\begin{aligned}
\underset{x\in\mathbb{R}^n, y_s\in\mathbb{R}^m}{\text{minimize}} \quad & c^T x + \frac{1}{n}\sum_{s=1}^{n} q_s^T y_s \\
\text{subject to} \quad & Ax = b \\
& T_s x + W y_s = h_s, \qquad s = 1,\dots,n \\
& x \geq 0,\ y_s \geq 0, \qquad s = 1,\dots,n
\end{aligned}
$$

Preliminaries - Sample average approximation

- Ω infinite
- ξ continuous random variable
- Sample n scenarios $\omega_s,\ s = 1, \dots, n$ independently from Ω

$$\begin{aligned} \underset{x \in \mathbb{R}^n, y_s \in \mathbb{R}^m}{\text{minimize}} \quad & c^T x + \frac{1}{n} \sum_{s=1}^{n} q_s^T y_s \\ \text{subject to} \quad & Ax = b \\ & T_s x + W y_s = h_s, \qquad s = 1, \dots, n \\ & x \geq 0,\ y_s \geq 0, \qquad s = 1, \dots, n \end{aligned}$$

- Asymptotic convergence as n goes to infinity

Preliminaries - Sample average approximation

- Ω infinite
- ξ continuous random variable
- Sample n scenarios $\omega_s,\ s = 1, \dots, n$ independently from Ω

$$
\begin{aligned}
\underset{x \in \mathbb{R}^n, y_s \in \mathbb{R}^m}{\text{minimize}} \quad & c^T x + \frac{1}{n} \sum_{s=1}^{n} q_s^T y_s \\
\text{subject to} \quad & Ax = b \\
& T_s x + W y_s = h_s, \qquad s = 1, \dots, n \\
& x \geq 0,\ y_s \geq 0, \qquad s = 1, \dots, n
\end{aligned}
$$

- Asymptotic convergence as n goes to infinity
- Confidence intervals around optimal solution

Preliminaries - Solution algorithms

All methods boil down to solving the finite extensive form

$$\begin{aligned} \underset{x\in\mathbb{R}^n, y_s\in\mathbb{R}^m}{\text{minimize}} \quad & c^T x + \sum_{s=1}^{n} \pi_s q_s^T y_s \\ \text{subject to} \quad & Ax = b \\ & T_s x + W y_s = h_s, \qquad s = 1, \dots, n \\ & x \geq 0,\ y_s \geq 0, \qquad s = 1, \dots, n \end{aligned}$$

Preliminaries - Solution algorithms

All methods boil down to solving the finite extensive form

$$\begin{aligned} \underset{x\in\mathbb{R}^n, y_s\in\mathbb{R}^m}{\text{minimize}} \quad & c^T x + \sum_{s=1}^{n} \pi_s q_s^T y_s \\ \text{subject to} \quad & Ax = b \\ & T_s x + W y_s = h_s, \qquad s = 1, \dots, n \\ & x \geq 0, \; y_s \geq 0, \qquad s = 1, \dots, n \end{aligned}$$

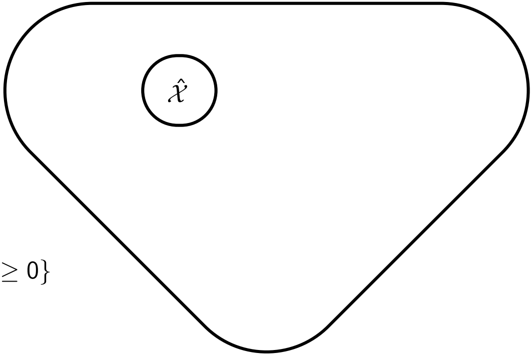
- Direct solution
- The L-shaped algorithm
- Progressive hedging

Preliminaries - Solution algorithms

$$\begin{pmatrix} A & & & & \\ T_1 & W & & & \\ T_2 & & W & & \\ \vdots & & & \ddots & \\ T_n & & & & W \end{pmatrix}$$

Figure: Stochastic program structure.

Preliminaries - The L-shaped algorithm

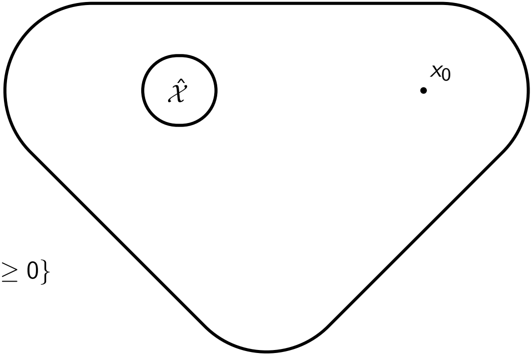


$$\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$$

$\hat{\mathcal{X}}$ = Optimal set

Figure: Cutting-plane method

Preliminaries - The L-shaped algorithm

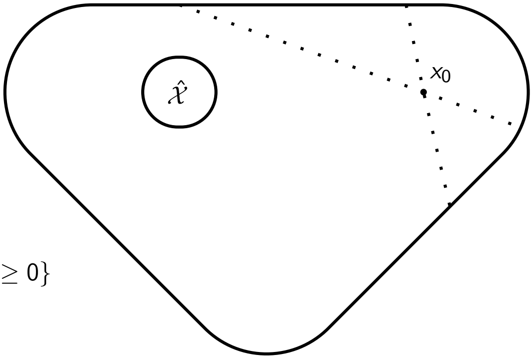


$$\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$$

$$\hat{\mathcal{X}} = \text{Optimal set}$$

Figure: Cutting-plane method

Preliminaries - The L-shaped algorithm

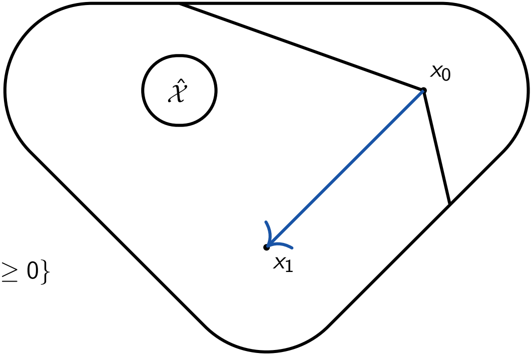


$$\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$$

$$\hat{\mathcal{X}} = \text{Optimal set}$$

Figure: Cutting-plane method

Preliminaries - The L-shaped algorithm

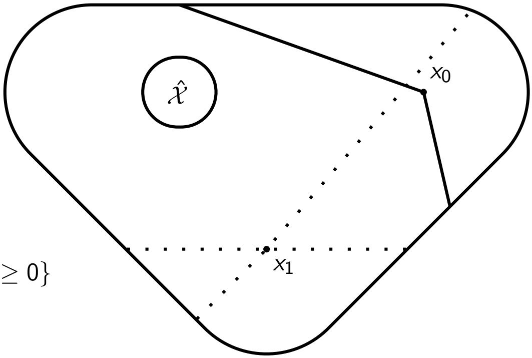


$$\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$$

$\hat{\mathcal{X}}$ = Optimal set

Figure: Cutting-plane method

Preliminaries - The L-shaped algorithm

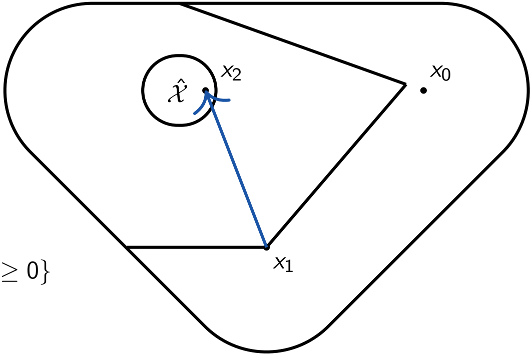


$\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$

$\hat{\mathcal{X}} =$ Optimal set

Figure: Cutting-plane method

Preliminaries - The L-shaped algorithm



$$\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$$

$$\hat{\mathcal{X}} = \text{Optimal set}$$

Figure: Cutting-plane method

Preliminaries - The L-shaped algorithm

Master problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x + \theta \\ \text{subject to} \quad & Ax = b \\ & \partial Q_k x + \theta \geq q_k, \quad \forall k \\ & x \geq 0 \end{aligned}$$

Subproblems

$$\begin{aligned} \underset{y_s \in \mathbb{R}^m}{\text{minimize}} \quad & Q_s^k = q_s^T y_s \\ \text{subject to} \quad & W y_s = h_s - T_s x_k \\ & y_s \geq 0 \end{aligned}$$

Optimality cuts

$$\partial Q_k = \sum_{s=1}^{n} \pi_s \lambda_s^T T_s, \quad q_k = \sum_{s=1}^{n} \pi_s \lambda_s^T h_s$$

Preliminaries - The L-shaped algorithm

Master problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x + \theta + \|x - \tilde{x}\| \\ \text{subject to} \quad & Ax = b \\ & \partial Q_k x + \theta \geq q_k, \qquad \forall k \\ & x \geq 0 \end{aligned}$$

Subproblems

$$\begin{aligned} \underset{y_s \in \mathbb{R}^m}{\text{minimize}} \quad & Q_s^k = q_s^T y_s \\ \text{subject to} \quad & W y_s = h_s - T_s x_k \\ & y_s \geq 0 \end{aligned}$$

Optimality cuts

$$\partial Q_k = \sum_{s=1}^{n} \pi_s \lambda_s^T T_s, \quad q_k = \sum_{s=1}^{n} \pi_s \lambda_s^T h_s$$

Preliminaries - The L-shaped algorithm

Master problem

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x + \sum_{s=1}^{n} \theta_s \\ \text{subject to} \quad & Ax = b \\ & \left.\begin{aligned} & \partial Q_{1,k} x + \theta_1 \geq q_{1,k}, \\ & \vdots \\ & \partial Q_{n,k} x + \theta_n \geq q_{n,k}, \end{aligned}\right\} \forall k \\ & x \geq 0 \end{aligned}$$

Subproblems

$$\begin{aligned} \underset{y_s \in \mathbb{R}^m}{\text{minimize}} \quad & Q_s^k = q_s^T y_s \\ \text{subject to} \quad & W y_s = h_s - T_s x_k \\ & y_s \geq 0 \end{aligned}$$

Optimality cuts

$$\partial Q_{s,k} = \pi_s \lambda_s^T T_s, \quad q_{s,k} = pi_s \lambda_s^T h_s$$

Preliminaries - The L-shaped algorithm

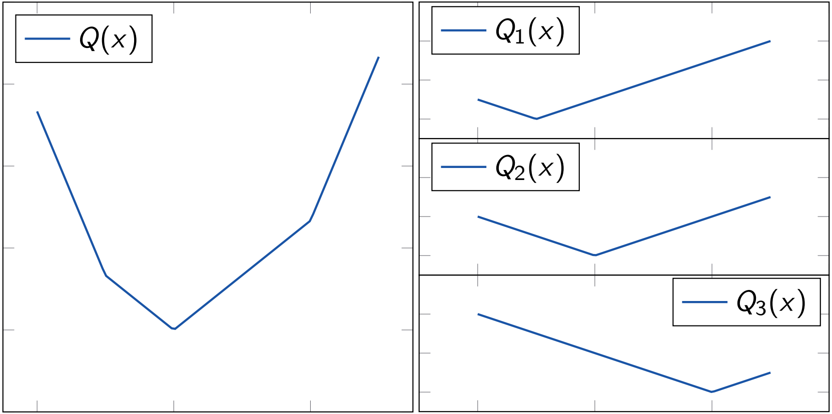


Figure: L-shaped procedure

Preliminaries - The L-shaped algorithm

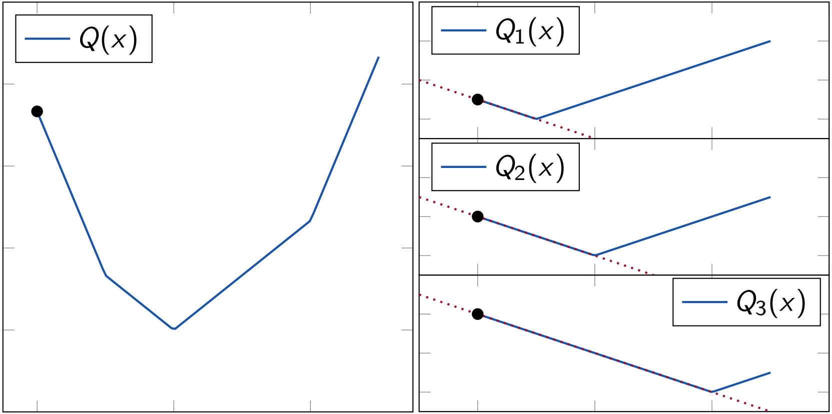


Figure: L-shaped procedure

Preliminaries - The L-shaped algorithm

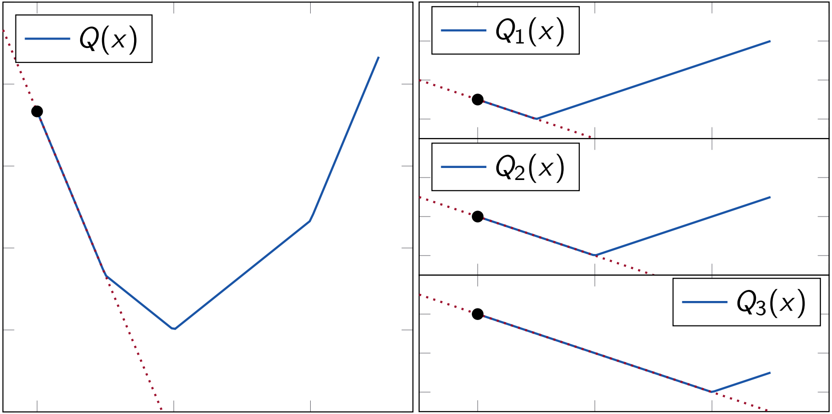


Figure: L-shaped procedure

Preliminaries - The L-shaped algorithm

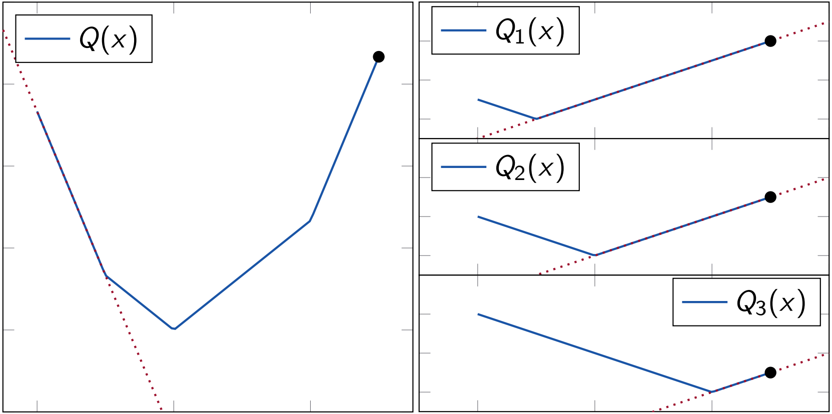


Figure: L-shaped procedure

Preliminaries - The L-shaped algorithm

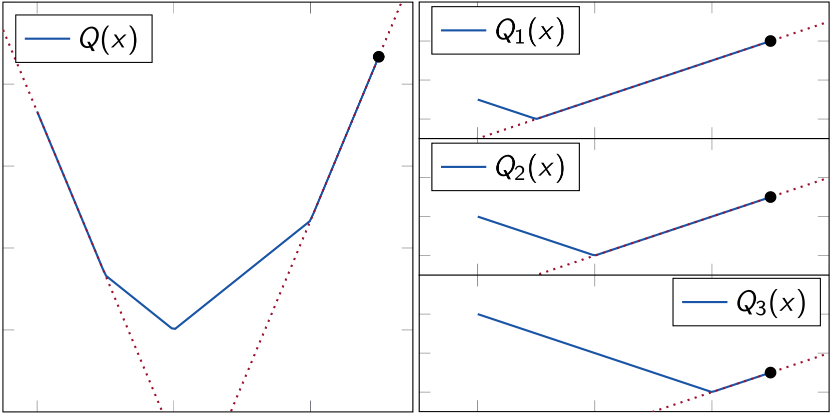


Figure: L-shaped procedure

Preliminaries - The L-shaped algorithm

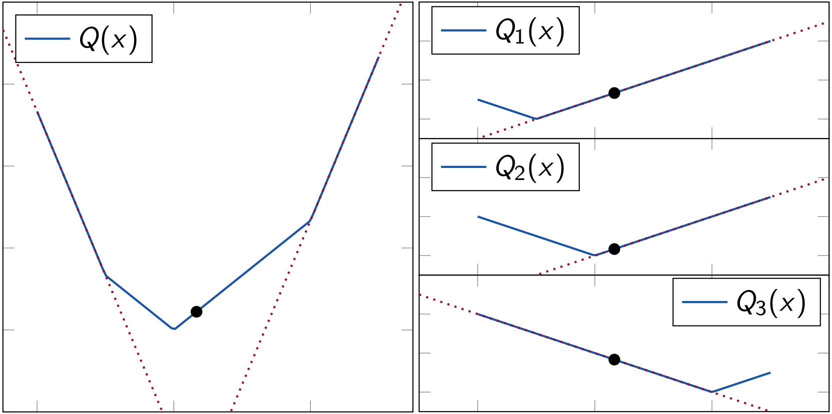


Figure: L-shaped procedure

Preliminaries - The L-shaped algorithm

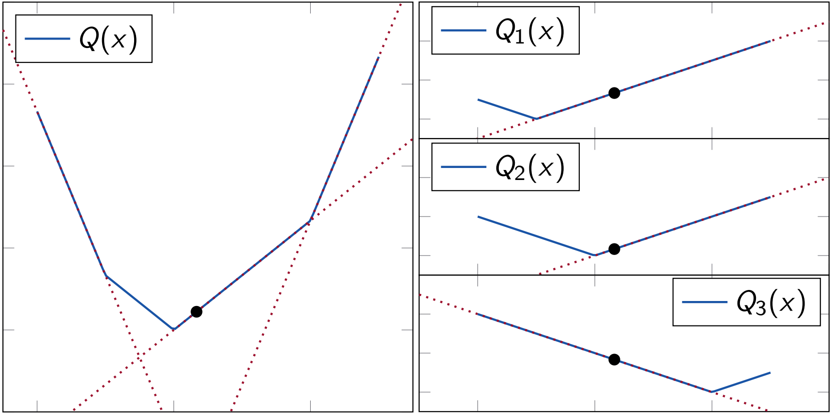


Figure: L-shaped procedure

Preliminaries - The L-shaped algorithm

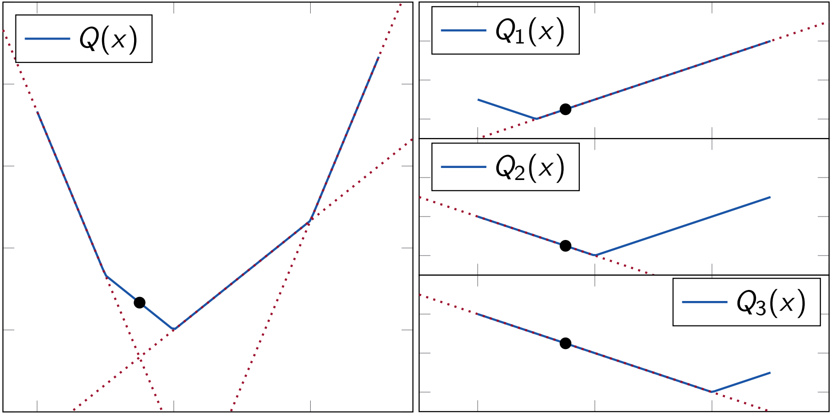


Figure: L-shaped procedure

Preliminaries - The L-shaped algorithm

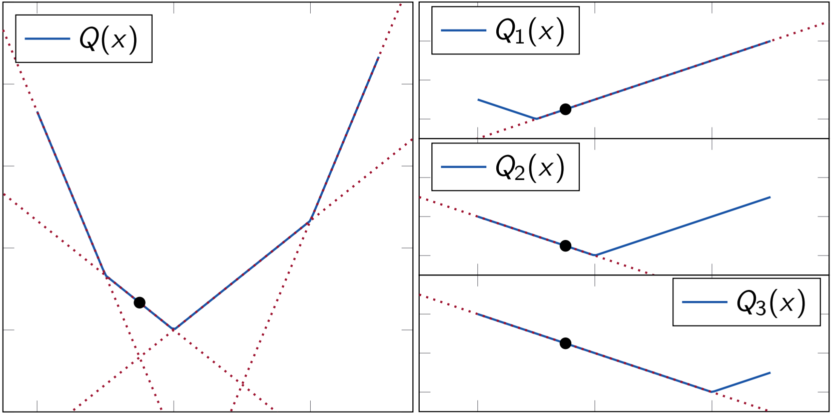


Figure: L-shaped procedure

Preliminaries - The L-shaped algorithm

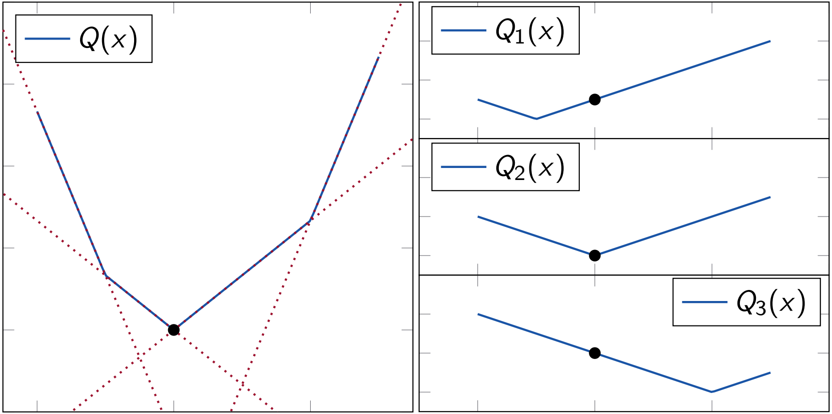


Figure: L-shaped procedure

Outline

Motivation

- Industry-scale applications typically involve 10,000+ scenarios

Motivation

- Industry-scale applications typically involve 10,000+ scenarios
- Example: 24-hour unit commitment problem [*Petra et al (2014)*]
  - 16,384 scenarios
  - 1.95 billion variables and constraints in the extended form
  - $\sim$ 1 hour computation time on a Titan supercomputer

Motivation

- Industry-scale applications typically involve 10,000+ scenarios
- Example: 24-hour unit commitment problem [*Petra et al (2014)*]
  - 16,384 scenarios
  - 1.95 billion variables and constraints in the extended form
  - $\sim$ 1 hour computation time on a Titan supercomputer
- Long computation time required to optimize

Motivation

- Industry-scale applications typically involve 10,000+ scenarios
- Example: 24-hour unit commitment problem [*Petra et al (2014)*]
  - 16,384 scenarios
  - 1.95 billion variables and constraints in the extended form
  - $\sim$ 1 hour computation time on a Titan supercomputer
- Long computation time required to optimize
- Memory requirement exceeds the capacity of a single machine

Motivation

- Industry-scale applications typically involve 10,000+ scenarios
- Example: 24-hour unit commitment problem [*Petra et al (2014)*]
  - 16,384 scenarios
  - 1.95 billion variables and constraints in the extended form
  - ~ 1 hour computation time on a Titan supercomputer
- Long computation time required to optimize
- Memory requirement exceeds the capacity of a single machine

Parallel algorithms that work on distributed data are required

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
- Distributed-memory implementation for large-scale models

Contribution

- `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

Contribution

- `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

Publications

- Martin Biel and Mikael Johansson. Efficient stochastic programming in Julia. *arXiv preprint arXiv:1909.10451*, 2019. Submitted for consideration to Mathematical Programming Computation. Under review,
- Martin Biel and Mikael Johansson. Distributed L-shaped algorithms in Julia. In *2018 IEEE/ACM Parallel Applications Workshop, Alternatives To MPI (PAW-ATM)*. IEEE, 2018.

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 model analysis
  - VSS
  - EVPI
  - Confidence intervals
  - ...

StochasticPrograms.jl

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

StochasticPrograms.jl

- Flexible and expressive problem definition
- Deferred model instantiation
- Scenario data injection
- Variety of tools for model analysis
  - 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 model analysis
  - 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

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

where

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

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

```
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

```
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

```
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

```
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
```

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

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
```

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

StochasticPrograms.jl - Simple model

```
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
```

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

StochasticPrograms.jl - Simple model

```
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
```

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

StochasticPrograms.jl - Discrete distribution

Let ξ have a discrete probability distribution, taking on the value

$$\xi_1 = \begin{pmatrix} 500 & 100 & -24 & -28 \end{pmatrix}^T$$

with probability 0.4 and

$$\xi_2 = \begin{pmatrix} 300 & 300 & -28 & -32 \end{pmatrix}^T$$

with probability 0.6.

StochasticPrograms.jl - Discrete distribution

```
ξ₁ = Scenario(q₁ = -24.0, q₂ = -28.0, d₁ = 500.0, d₂ = 100.0, probability = 0.4);
ξ₂ = Scenario(q₁ = -28.0, q₂ = -32.0, d₁ = 300.0, d₂ = 300.0, probability = 0.6);
sp = instantiate(simple_model, [ξ₁,ξ₂])

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

StochasticPrograms.jl - Discrete distribution

```
ξ₁ = Scenario(q₁ = -24.0, q₂ = -28.0, d₁ = 500.0, d₂ = 100.0, probability = 0.4);
ξ₂ = Scenario(q₁ = -28.0, q₂ = -32.0, d₁ = 300.0, d₂ = 300.0, probability = 0.6);
sp = instantiate(simple_model, [ξ₁,ξ₂])

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

StochasticPrograms.jl - Discrete distribution

```
ξ₁ = Scenario(q₁ = -24.0, q₂ = -28.0, d₁ = 500.0, d₂ = 100.0, probability = 0.4);
ξ₂ = Scenario(q₁ = -28.0, q₂ = -32.0, d₁ = 300.0, d₂ = 300.0, probability = 0.6);
sp = instantiate(simple_model, [ξ₁, ξ₂])

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

StochasticPrograms.jl - Discrete distribution

```
print(sp)

 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(sp)
print(dep)
```

$$
\begin{aligned}
&\text{Min } 100\, x_1 + 150\, x_2 - 9.6\, y_{11} - 11.2\, y_{21} - 16.8\, y_{12} - 19.2\, y_{22} \\
&\text{Subject to} \\
&\quad x_1 + x_2 \leq 120 \\
&\quad 6\, y_{11} + 10\, y_{21} - 60\, x_1 \leq 0 \\
&\quad 8\, y_{11} + 5\, y_{21} - 80\, x_2 \leq 0 \\
&\quad 6\, y_{12} + 10\, y_{22} - 60\, x_1 \leq 0 \\
&\quad 8\, y_{12} + 5\, y_{22} - 80\, x_2 \leq 0 \\
&\quad x_1 \geq 40 \\
&\quad x_2 \geq 20 \\
&\quad 0 \leq y_{11} \leq 500 \\
&\quad 0 \leq y_{21} \leq 100 \\
&\quad 0 \leq y_{12} \leq 300 \\
&\quad 0 \leq y_{22} \leq 300
\end{aligned}
$$

StochasticPrograms.jl - Discrete distribution

```
dep = DEP(sp)
print(dep)
```

Min $100\ x_1 + 150\ x_2 - 9.6\ y_{11} - 11.2\ y_{21} - 16.8\ y_{12} - 19.2\ y_{22}$
Subject to
$x_1 + x_2 \leq 120$
$6\ y_{11} + 10\ y_{21} - 60\ x_1 \leq 0$
$8\ y_{11} + 5\ y_{21} - 80\ x_2 \leq 0$
$6\ y_{12} + 10\ y_{22} - 60\ x_1 \leq 0$
$8\ y_{12} + 5\ y_{22} - 80\ x_2 \leq 0$
$x_1 \geq 40$
$x_2 \geq 20$
$0 \leq y_{11} \leq 500$
$0 \leq y_{21} \leq 100$
$0 \leq y_{12} \leq 300$
$0 \leq y_{22} \leq 300$

StochasticPrograms.jl - Discrete distribution

```
dep = DEP(sp)
print(dep)
```

$$\begin{aligned}
&\text{Min } 100\, x_1 + 150\, x_2 - 9.6\, y_{11} - 11.2\, y_{21} - 16.8\, y_{12} - 19.2\, y_{22} \\
&\text{Subject to} \\
&\; x_1 + x_2 \leq 120 \\
&\; 6\, y_{11} + 10\, y_{21} - 60\, x_1 \leq 0 \\
&\; 8\, y_{11} + 5\, y_{21} - 80\, x_2 \leq 0 \\
&\; 6\, y_{12} + 10\, y_{22} - 60\, x_1 \leq 0 \\
&\; 8\, y_{12} + 5\, y_{22} - 80\, x_2 \leq 0 \\
&\; x_1 \geq 40 \\
&\; x_2 \geq 20 \\
&\; 0 \leq y_{11} \leq 500 \\
&\; 0 \leq y_{21} \leq 100 \\
&\; 0 \leq y_{12} \leq 300 \\
&\; 0 \leq y_{22} \leq 300
\end{aligned}$$

StochasticPrograms.jl - Discrete distribution

```
vrp = VRP(sp, solver = gurobi) # value of the recourse problem
-855.83

vss = VSS(sp, solver = gurobi) # value of the stochastic solution
286.92
```

StochasticPrograms.jl - Discrete distribution

```
vrp = VRP(sp, solver = gurobi) # value of the recourse problem
-855.83

vss = VSS(sp, solver = gurobi) # value of the stochastic solution
286.92
```

StochasticPrograms.jl - Discrete distribution

```
vrp = VRP(sp, solver = gurobi) # value of the recourse problem
-855.83

vss = VSS(sp, solver = gurobi) # value of the stochastic solution
286.92
```

StochasticPrograms.jl - Continuous distribution

Let instead ξ have a multivariate normal distribution $\xi \sim \mathcal{N}(\mu, \Sigma)$, where

$$\mu = \begin{pmatrix} -28 \\ -32 \\ 300 \\ 300 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} 2 & 0.5 & 0 & 0 \\ 0.5 & 1 & 0 & 0 \\ 0 & 0 & 50 & 20 \\ 0 & 0 & 20 & 30 \end{pmatrix}$$

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

```
@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

```
@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

```
@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

```
@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

```
@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

```
@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
```

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]
```

StochasticPrograms.jl - Solvers

```
optimize!(sp, solver = GurobiSolver())
optimal_value(sp)
-855.83
```

StochasticPrograms.jl - Solvers

```
optimize!(sp, solver = GurobiSolver())
optimal_value(sp)
-855.83
```

```
optimize!(sp, solver = LShapedSolver(gurobi))
L-Shaped Gap  Time: 0:00:00 (6 iterations)
  Objective:       -855.8333
  Gap:             0.0
  No. cuts:        7
  Iterations:      6
```

StochasticPrograms.jl - Solvers

```
optimize!(sp, solver = GurobiSolver())
optimal_value(sp)
-855.83
```

```
optimize!(sp, solver = LShapedSolver(gurobi))
L-Shaped Gap  Time: 0:00:00 (6 iterations)
  Objective:       -855.8333
  Gap:             0.0
  No. cuts:        7
  Iterations:      6
```

```
optimize!(sp, solver = ProgressiveHedgingSolver(gurobi))
Progressive Hedging Time: 0:00:06 (1315 iterations)
  Objective:   -855.8333
  δ:           9.570267362791345e-7
```

StochasticPrograms.jl - Distributed models

```
using Distributed
addprocs(2)
...
sp = instantiate(simple_model, [ξ₁, ξ₂])
Distributed stochastic program with:
 * 2 decision variables
 * 2 recourse variables
 * 2 scenarios of type Scenario
Solver is default solver
```

StochasticPrograms.jl - Distributed models

```
using Distributed
addprocs(2)
...
sp = instantiate(simple_model, [ξ₁, ξ₂])
Distributed stochastic program with:
 * 2 decision variables
 * 2 recourse variables
 * 2 scenarios of type Scenario
Solver is default solver
```

```
optimize!(sp, solver = LShapedSolver(gurobi, distributed = true))
Distributed L-Shaped Gap  (thresh = 1e-06, value = 0.0)
  Objective:       -855.833
  Gap:             0.0
  No. cuts:        5
  Iterations:      4
```

StochasticPrograms.jl - Implementation

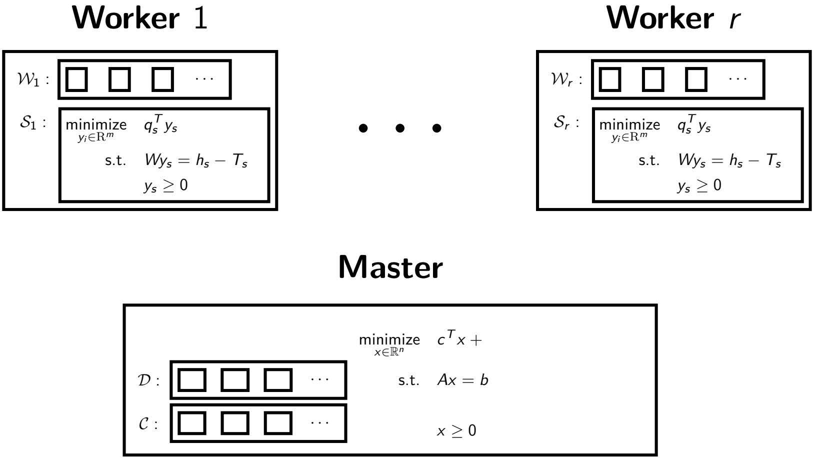


Figure: Distributed L-shaped procedure

StochasticPrograms.jl - Implementation

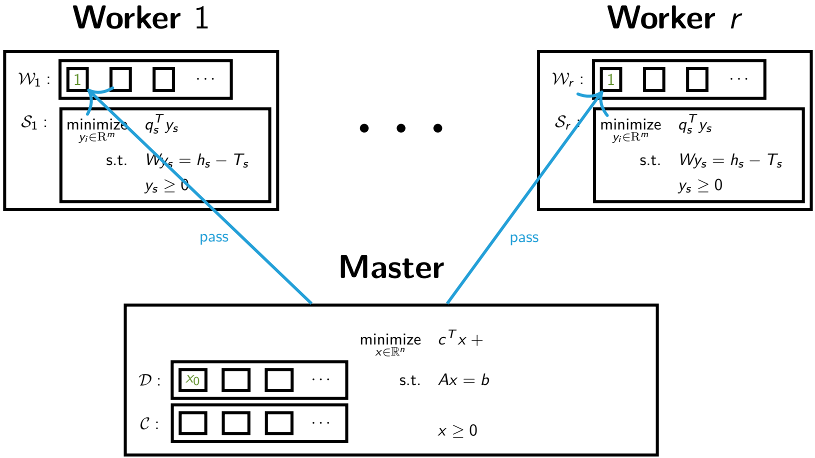


Figure: Distributed L-shaped procedure

StochasticPrograms.jl - Implementation

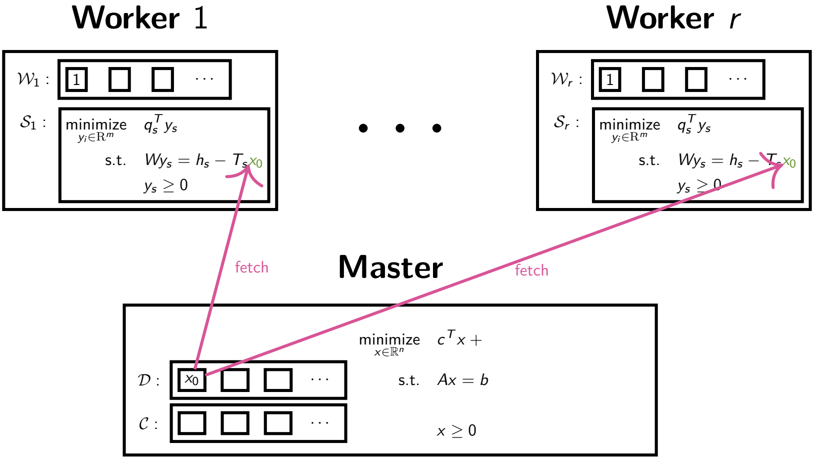


Figure: Distributed L-shaped procedure

StochasticPrograms.jl - Implementation

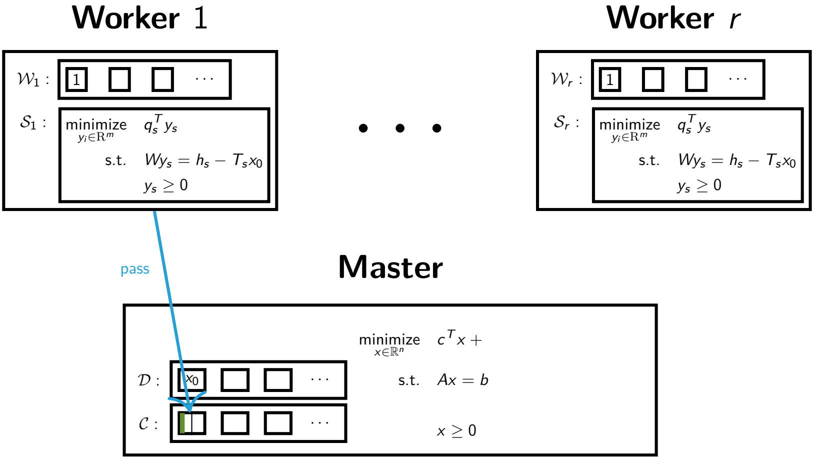


Figure: Distributed L-shaped procedure

StochasticPrograms.jl - Implementation

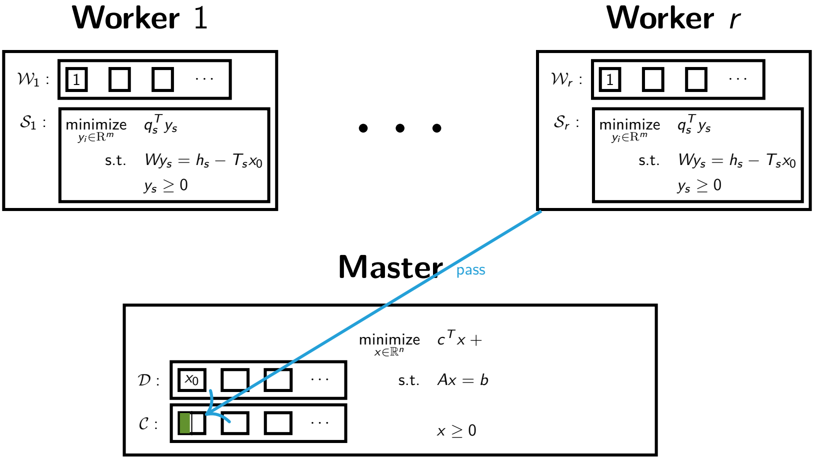


Figure: Distributed L-shaped procedure

StochasticPrograms.jl - Implementation

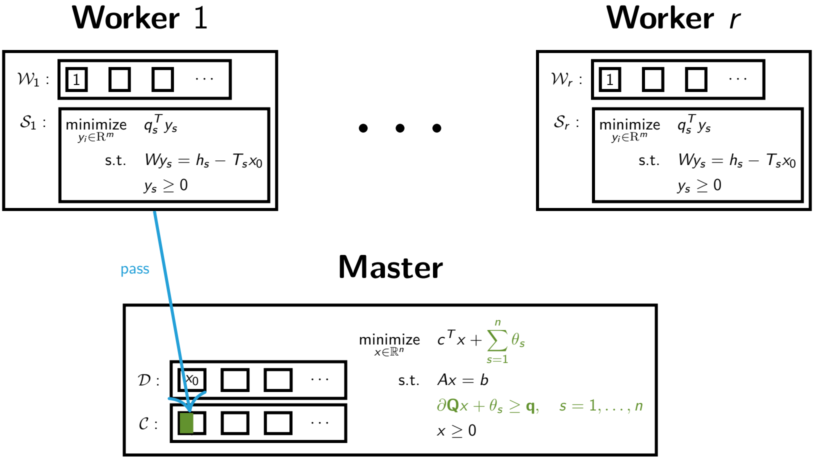


Figure: Distributed L-shaped procedure

StochasticPrograms.jl - Implementation

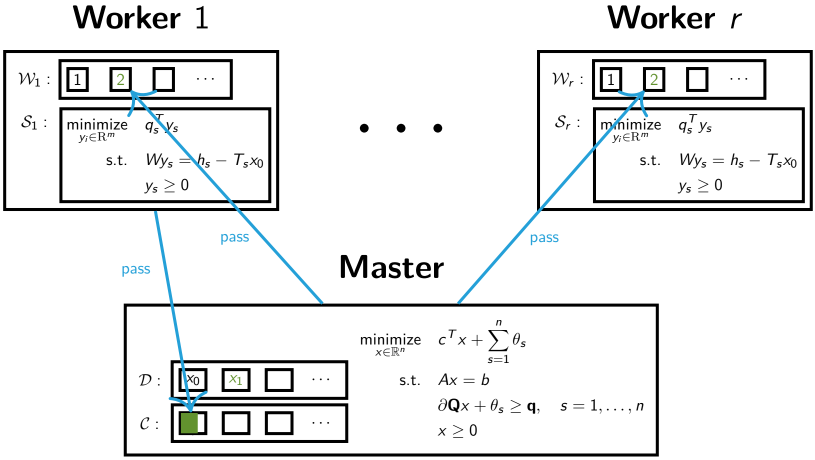


Figure: Distributed L-shaped procedure

StochasticPrograms.jl - Implementation

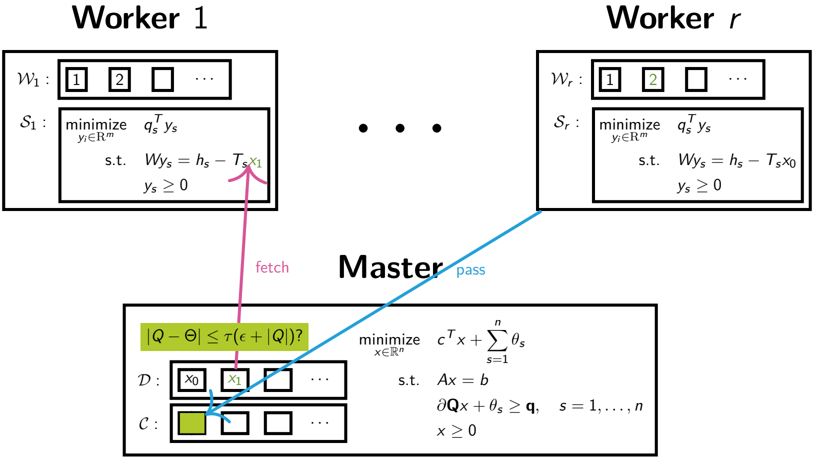


Figure: Distributed L-shaped procedure

StochasticPrograms.jl - Implementation

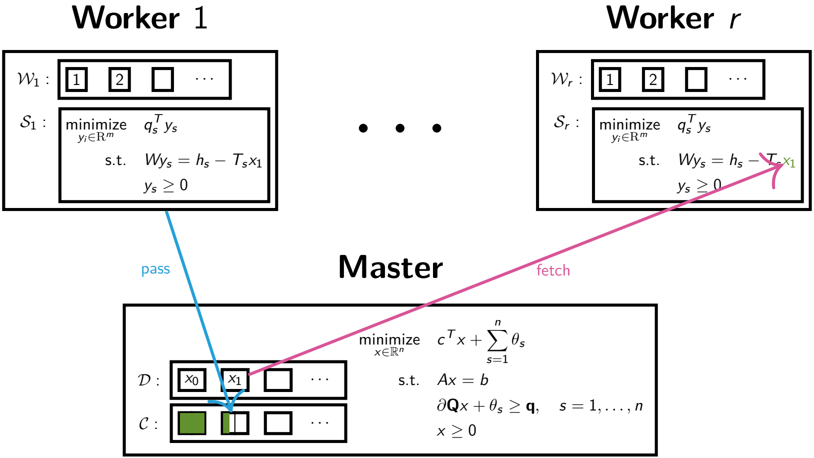


Figure: Distributed L-shaped procedure

StochasticPrograms.jl - Numerical experiments

The day-ahead problem

- Optimal order strategies on a deregulated electricity market
- From the perspective of a hydropower producer
- First stage: Hourly electricity volume bids for the upcoming day
- Second stage: Optimize production when market price is known

StochasticPrograms.jl - Numerical experiments

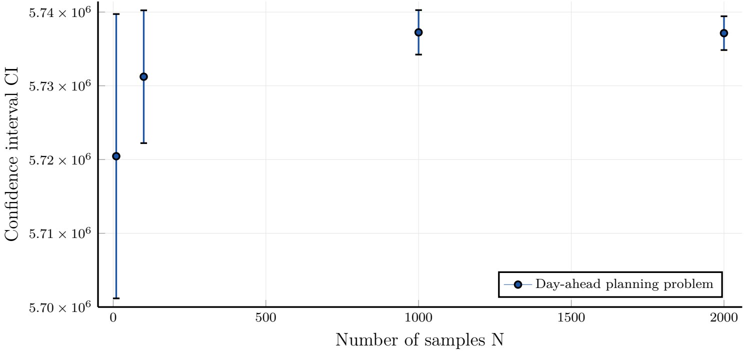


Figure: Confidence intervals around optimal value of the day-ahead problem as a function of SAA sample size.

StochasticPrograms.jl - Numerical experiments

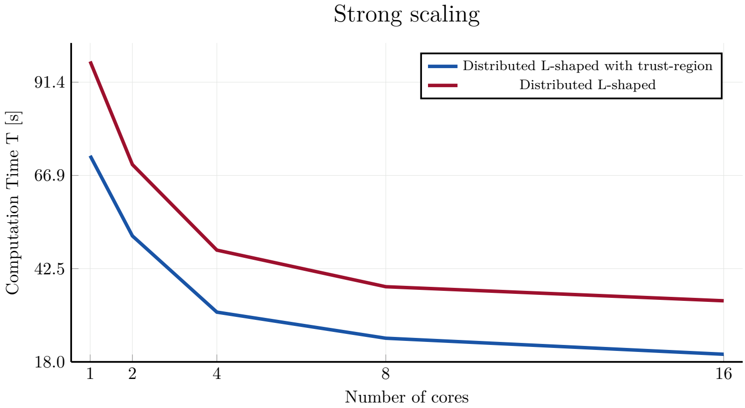


Figure: Median computation time required for L-shaped algorithms to solve a day-ahead problem with 1000 scenarios, as a function of number of worker cores.

StochasticPrograms.jl - Numerical experiments

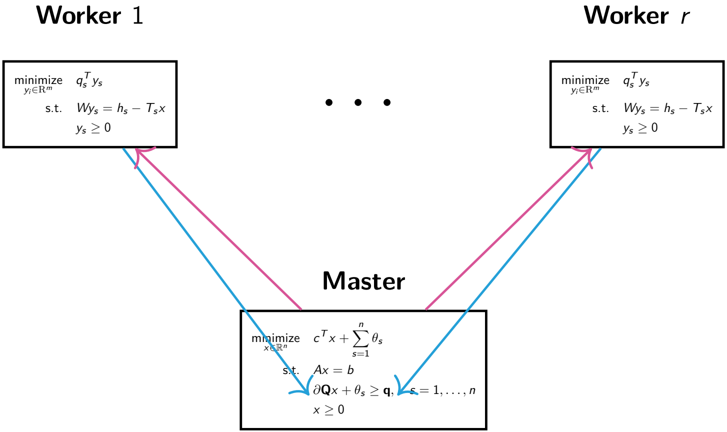


Figure: Load imbalance in distributed L-shaped procedure

StochasticPrograms.jl - Numerical experiments

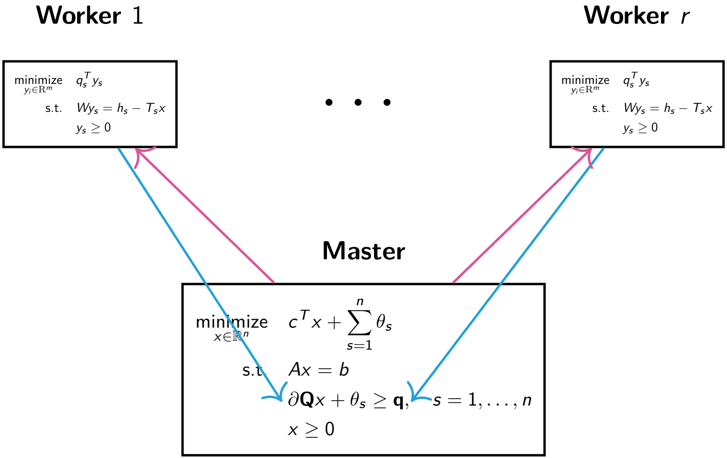


Figure: Load imbalance in distributed L-shaped procedure

StochasticPrograms.jl - Numerical experiments

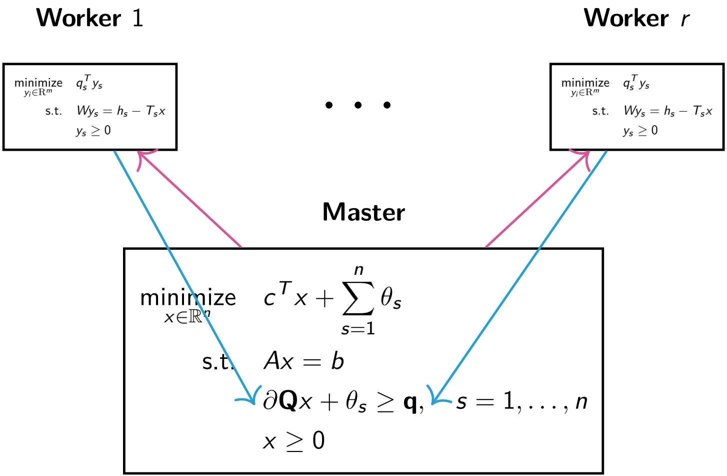


Figure: Load imbalance in distributed L-shaped procedure

StochasticPrograms.jl - Numerical experiments

Cut aggregation

StochasticPrograms.jl - Numerical experiments

Cut aggregation

- Partition optimality cuts into uniform aggregates

StochasticPrograms.jl - Numerical experiments

Cut aggregation

- Partition optimality cuts into uniform aggregates
- $\partial Q_{a,k} = \sum_{s \in \mathcal{S}_a} \pi_s \lambda_s^T T_s, \quad q_{a,k} = \sum_{s \in \mathcal{S}_a} \pi_s \lambda_s^T h_s$

StochasticPrograms.jl - Numerical experiments

Cut aggregation

- Partition optimality cuts into uniform aggregates
- $\partial Q_{a,k} = \sum_{s \in \mathcal{S}_a} \pi_s \lambda_s^T T_s, \quad q_{a,k} = \sum_{s \in \mathcal{S}_a} \pi_s \lambda_s^T h_s$
- Reduce amount of passed data

StochasticPrograms.jl - Numerical experiments

Cut aggregation

- Partition optimality cuts into uniform aggregates
- $\partial Q_{a,k} = \sum_{s \in \mathcal{S}_a} \pi_s \lambda_s^T T_s, \quad q_{a,k} = \sum_{s \in \mathcal{S}_a} \pi_s \lambda_s^T h_s$
- Reduce amount of passed data
- Master problem does not grow as fast

StochasticPrograms.jl - Numerical experiments

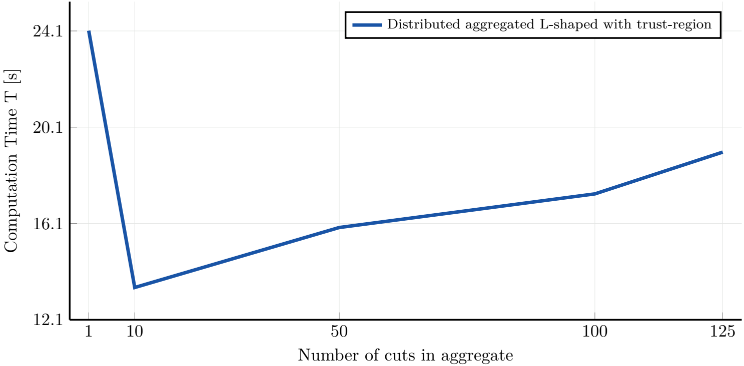


Figure: Median computation time required for the aggregated L-shaped method to solve a day-ahead problem with 1000 scenarios. The experiment was performed on 8 worker cores.

StochasticPrograms.jl - Summary

- `StochasticPrograms.jl`: framework for stochastic programming

StochasticPrograms.jl - Summary

- `StochasticPrograms.jl`: framework for stochastic programming
- Formulate and solve memory-distributed stochastic programs

StochasticPrograms.jl - Summary

- `StochasticPrograms.jl`: framework for stochastic programming
- Formulate and solve memory-distributed stochastic programs
- Structure-exploiting algorithms that run in parallel on distributed data

StochasticPrograms.jl - Summary

- `StochasticPrograms.jl`: framework for stochastic programming
- Formulate and solve memory-distributed stochastic programs
- Structure-exploiting algorithms that run in parallel on distributed data
- The full framework is open-source and freely available on Github

`https://github.com/martinbiel`

Outline

Motivation

- Cut aggregation improves distributed performance

Motivation

- Cut aggregation improves distributed performance
  - Reduce communication latency
  - Reduce load imbalance

Motivation

- Cut aggregation improves distributed performance
  - Reduce communication latency
  - Reduce load imbalance
- Uniform cut aggregation has been applied in many recent works

Motivation

- Cut aggregation improves distributed performance
  - Reduce communication latency
  - Reduce load imbalance
- Uniform cut aggregation has been applied in many recent works
- Complexity analysis only covers single-cut and multi-cut L-shaped

Contribution

- Review of the use of cut aggregation in L-shaped algorithms

Contribution

- Review of the use of cut aggregation in L-shaped algorithms
- Novel dynamic cut aggregation procedure

Contribution

- Review of the use of cut aggregation in L-shaped algorithms
- Novel dynamic cut aggregation procedure
- Theoretical results that complement earlier works

Contribution

- Review of the use of cut aggregation in L-shaped algorithms
- Novel dynamic cut aggregation procedure
- Theoretical results that complement earlier works
- Performance improvements in large-scale examples

Contribution

- Review of the use of cut aggregation in L-shaped algorithms
- Novel dynamic cut aggregation procedure
- Theoretical results that complement earlier works
- Performance improvements in large-scale examples

Publications

- Martin Biel and Mikael Johansson. Dynamic cut aggregation in L-shaped algorithms. *arXiv preprint arXiv:1910.13752*, 2019.
  Submitted for consideration to the European Journal of Operational Research. Under review

Cut aggregation - Worst case analysis

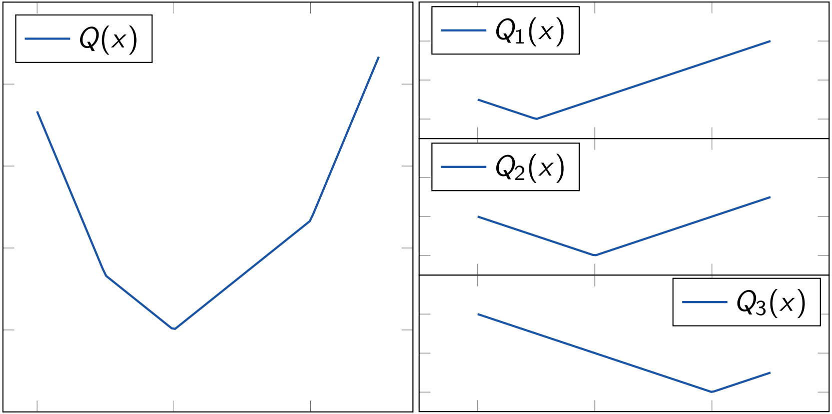


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

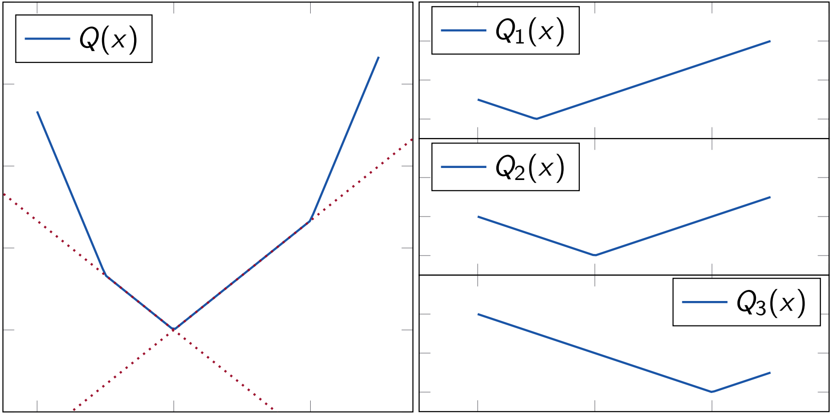


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

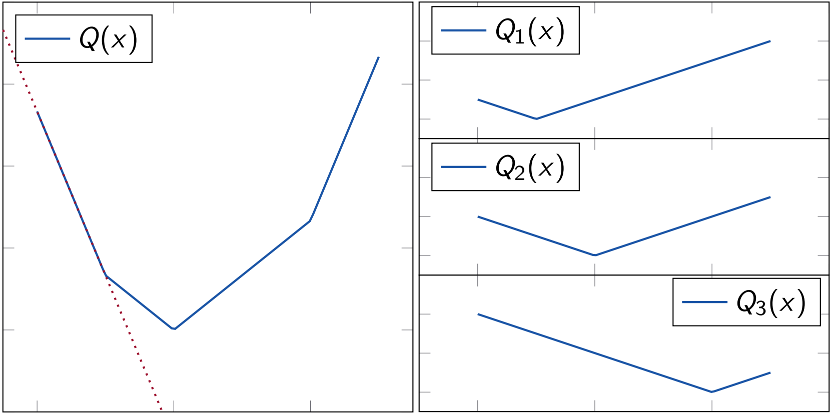


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

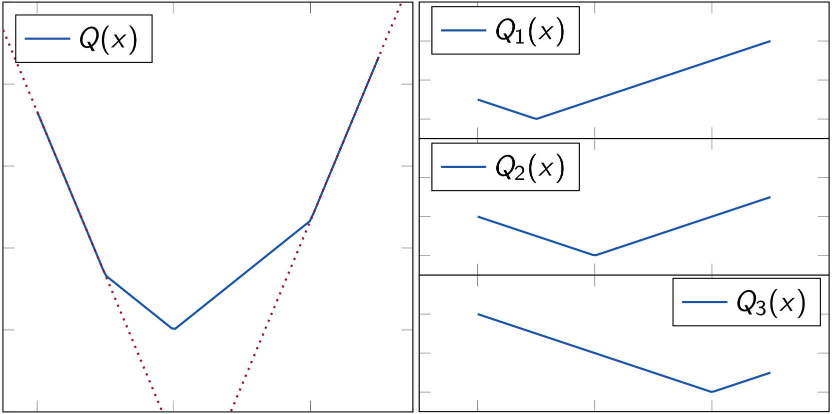


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

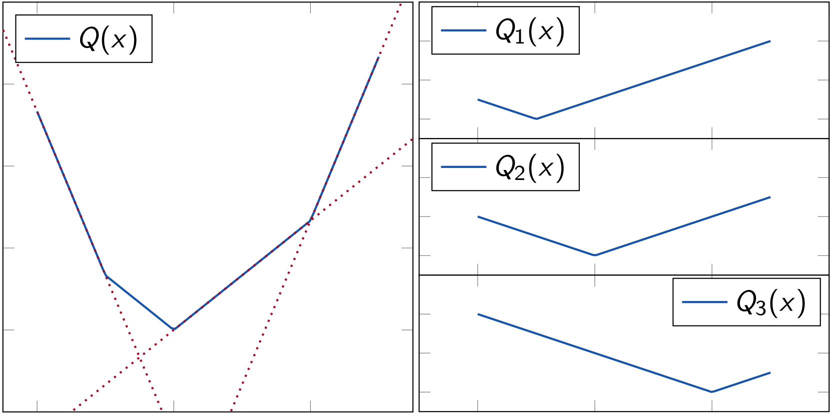


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

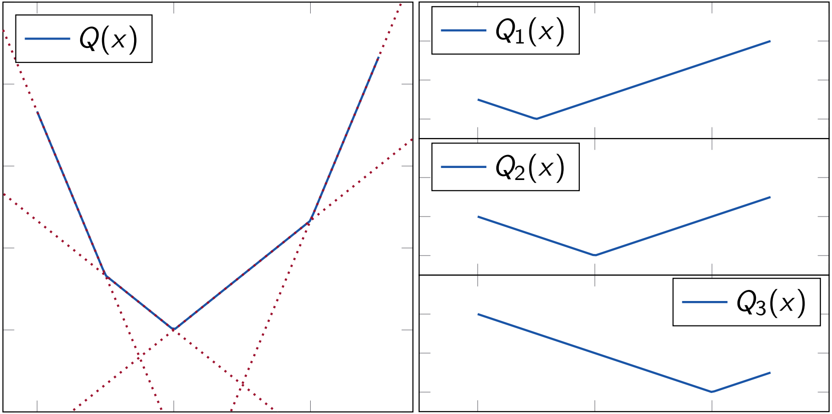


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

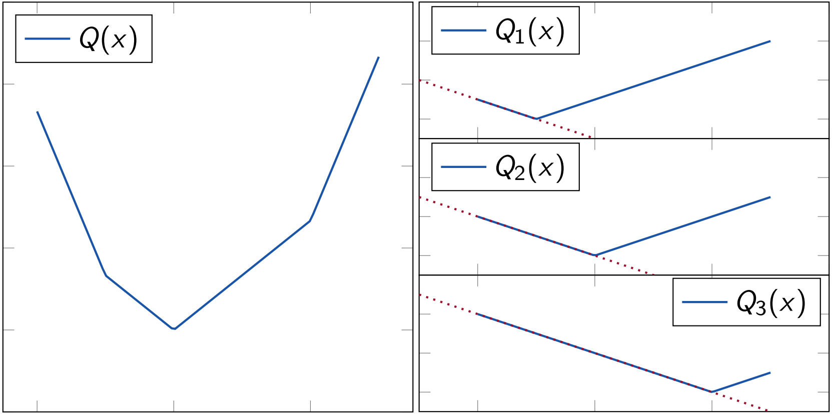


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

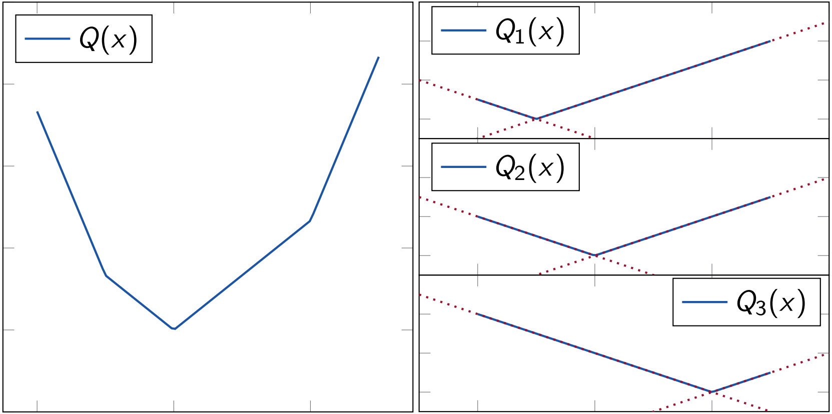


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

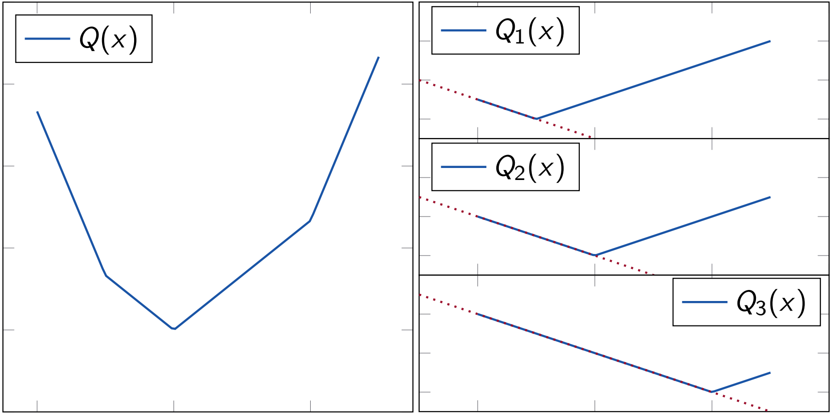


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

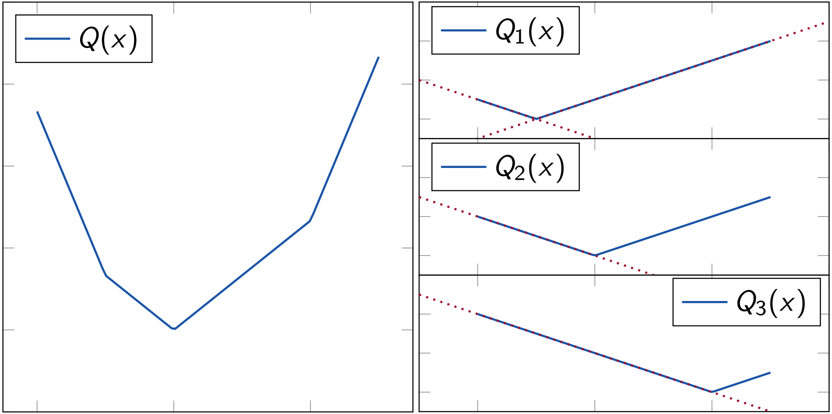


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

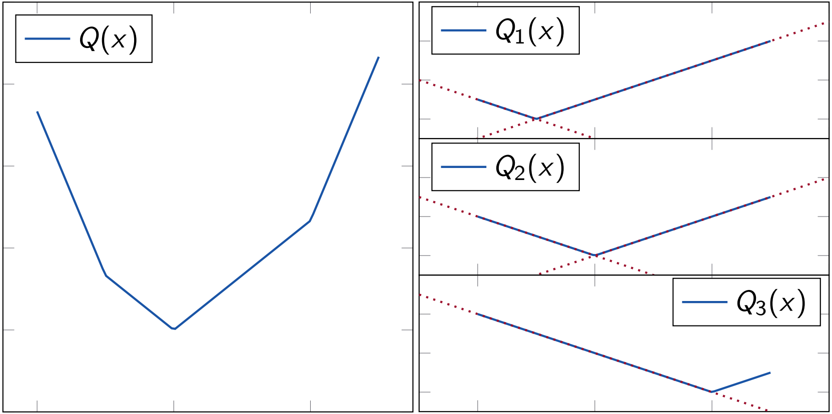


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

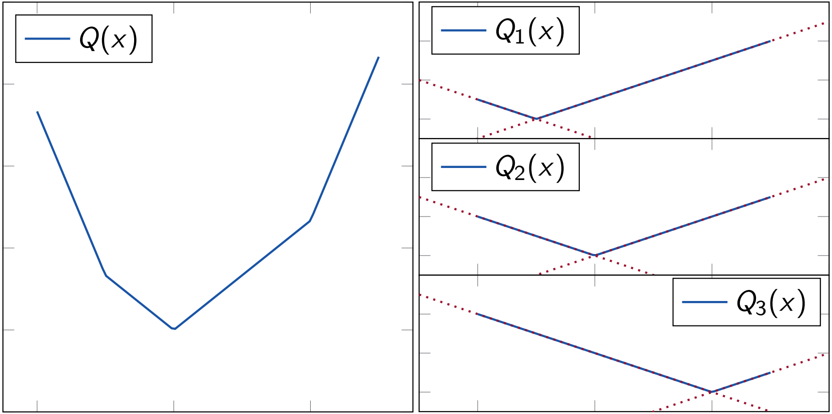


Figure: L-shaped procedure

Cut aggregation - Worst case analysis

Definition

Let b_s represent the maximum number of different slopes of $Q_s(x)$ in any direction parallel to one of the axes. Then, $b = \max_s b_s$ is the *slope number* of $Q(x)$.

Cut aggregation - Worst case analysis

Definition

Let b_s represent the maximum number of different slopes of $Q_s(x)$ in any direction parallel to one of the axes. Then, $b = \max_s b_s$ is the *slope number* of $Q(x)$.

Theorem (Birge and Louveaux, 1988)

The maximum number of iterations required to obtain an optimal solution is, for single-cut L-shaped:

$$[1 + n(b - 1)]^m,$$

and for multi-cut L-shaped:

$$1 + n(b^m - 1).$$

Static cut aggregation

Definition

A *partitioning scheme*

$$\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_A\}$$

of n scenarios is a set of partitions such that

$$\begin{aligned} \mathcal{S}_a &\subseteq \{1, \dots, n\}, & a &= 1, \dots, A \\ \mathcal{S}_a \cap \mathcal{S}_b &= \emptyset, & \forall a &\neq b \\ \bigcup_{a=1}^{A} \mathcal{S}_a &= \{1, \dots, n\}. \end{aligned}$$

Static cut aggregation

Aggregated L-shaped master

$$
\begin{aligned}
\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x + \sum_{a=1}^{A} \theta_a \\
\text{s.t.} \quad & Ax = b \\
& \partial Q_{a,k} x + \theta_a \geq q_{a,k}, \qquad a = 1, \dots, A \quad \forall k \\
& x \geq 0
\end{aligned}
$$

Aggregated optimality cuts

$$
\begin{aligned}
\partial Q_{a,k} &= \sum_{s \in \mathcal{S}_a} \pi_s \lambda_s^T T_s \\
q_{a,k} &= \sum_{s \in \mathcal{S}_a} \pi_s \lambda_s^T h_s
\end{aligned}
$$

Static cut aggregation

Definition

The *aggregation size* of the partitioning scheme $\mathcal{S}$ is given by

$$A(\mathcal{S}) = |\mathcal{S}|.$$

Definition

The *aggregation level* of the partitioning scheme $\mathcal{S}$ is given by

$$A_L(\mathcal{S}) = \max_{a=1,\dots,A(\mathcal{S})} |\mathcal{S}_a|.$$

Static cut aggregation

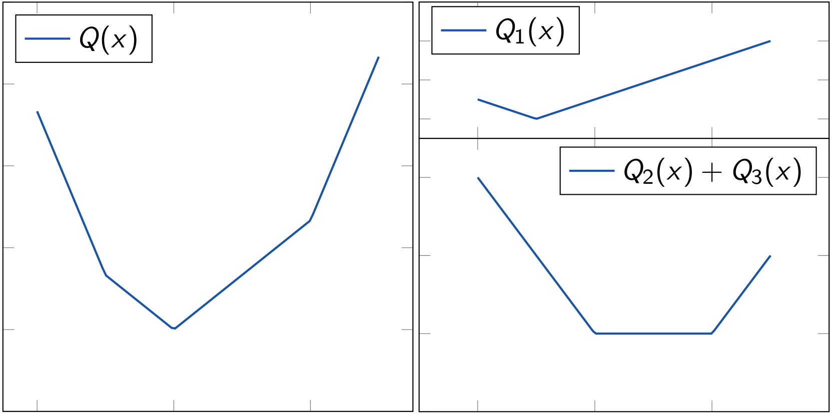


Figure: L-shaped with static aggregation

Static cut aggregation

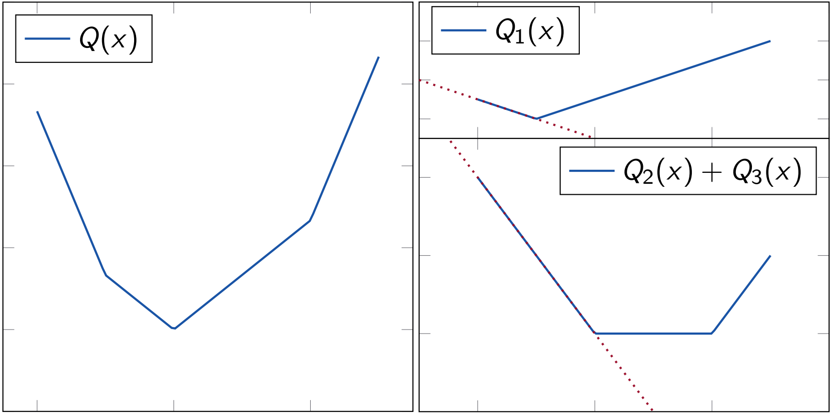


Figure: L-shaped with static aggregation

Static cut aggregation

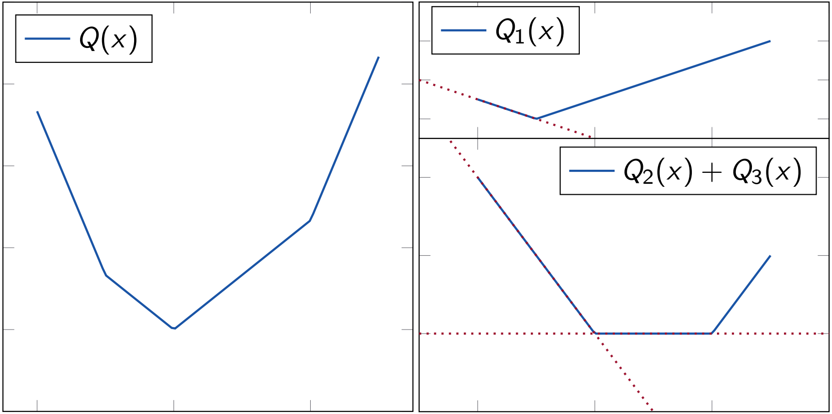


Figure: L-shaped with static aggregation

Static cut aggregation

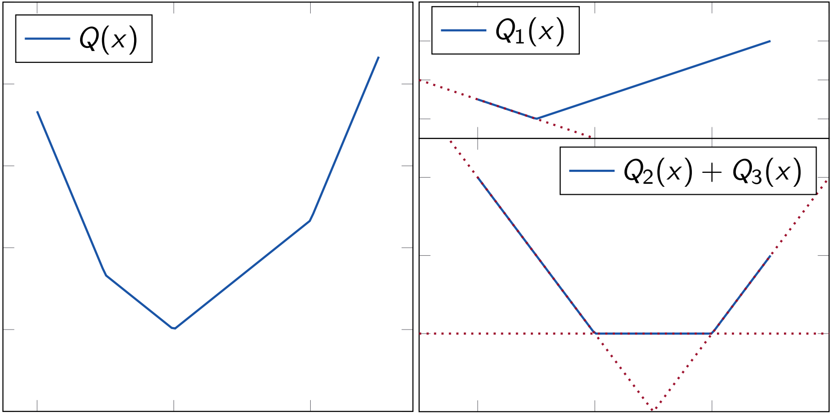


Figure: L-shaped with static aggregation

Static cut aggregation

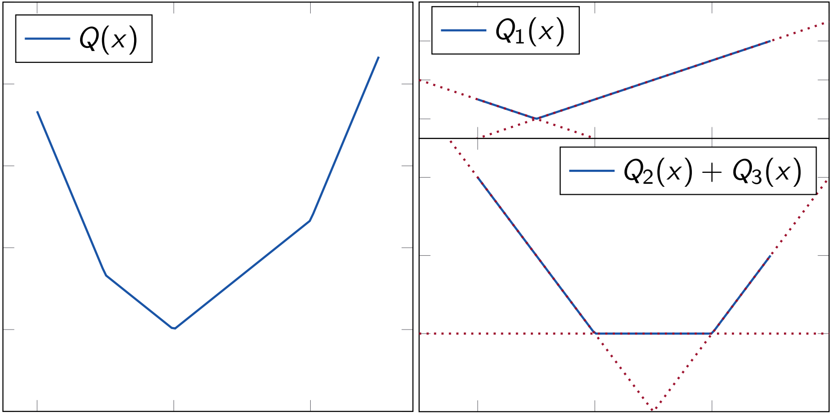


Figure: L-shaped with static aggregation

Static cut aggregation

Theorem

The maximum number of iterations required to obtain an optimal solution, of an aggregated L-shaped algorithm that uses a partitioning scheme $\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_{A(\mathcal{S})}\}$*, is given by*

$$1 + \sum_{a=1}^{A(\mathcal{S})} [1 + |\mathcal{S}_a|(b-1)]^m - A(\mathcal{S}).$$

Static cut aggregation

Corollary

The maximum number of iterations of an aggregated L-shaped algorithm, using a partitioning scheme $\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_A\}$*, is upper bounded by*

$$1 + A(\mathcal{S})([1 + A_L(\mathcal{S})(b-1)]^m - 1).$$

Static cut aggregation

Corollary

The maximum number of iterations of an aggregated L-shaped algorithm, using a partitioning scheme $\mathcal{S} = \{\mathcal{S}_1, \dots, \mathcal{S}_A\}$, *is upper bounded by*

$$1 + A(\mathcal{S})([1 + A_L(\mathcal{S})(b-1)]^m - 1).$$

Uniform cut aggregation

$A_L(\mathcal{S}) = n/A(\mathcal{S})$. Hence, worst case is given by

$$\frac{n^m(b-1)^m}{A(\mathcal{S})^{m-1}} < n^m(b-1)^m$$

Dynamic cut aggregation

Definition

A *dynamic partitioning scheme*

$$\mathcal{D} = \{\mathcal{S}^k\}_{k=1}^{\infty}$$

is a sequence of partitioning schemes $\mathcal{S}^k = \{\mathcal{S}_1^k, \dots, \mathcal{S}_{A_k}^k\}$.

Dynamic cut aggregation

Dynamically aggregated L-shaped master

$$
\begin{aligned}
\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad & c^T x + \sum_{s=1}^{n} \theta_s \\
\text{s.t.} \quad & Ax = b \\
& \sum_{s \in \mathcal{S}_a^k} \partial Q_{k,s} x + \sum_{s \in \mathcal{S}_a^k} \theta_s \geq \sum_{s \in \mathcal{S}_a^k} q_{s,k}, \qquad \mathcal{S}^k \in \mathcal{D} \quad \forall k \\
& x \geq 0.
\end{aligned}
$$

Dynamic cut aggregation

Dynamically aggregated L-shaped master

$$
\begin{aligned}
\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad & c^T x + \sum_{s=1}^{n} \theta_s \\
\text{s.t.} \quad & Ax = b \\
& \sum_{s\in\mathcal{S}_a^k} \partial Q_{k,s} x + \sum_{s\in\mathcal{S}_a^k} \theta_s \geq \sum_{s\in\mathcal{S}_a^k} q_{s,k}, \qquad \mathcal{S}^k \in \mathcal{D} \quad \forall k \\
& x \geq 0.
\end{aligned}
$$

Theorem

An L-shaped algorithm that uses dynamic cut aggregation, with a dynamic partitioning scheme $\mathcal{D} = \{\mathcal{S}^k\}_{k=1}^{\infty}$ converges to an optimal solution of a given stochastic program in a finite number of iterations.

Dynamic cut aggregation

Complexity

Theorem

The maximum number of iterations required to obtain an optimal solution, of an L-shaped algorithm that uses dynamic cut aggregation with a dynamic partitioning scheme $\mathcal{D} = \{\mathcal{S}^k\}_{k=1}^{\infty}$, *is given by*

$$2 + \sum_{a_L=1}^{n} \binom{n}{a_L} [1 + a_L(b-1)]^m - \sum_{a_L=1}^{n} \left\{ n \atop a_L \right\} - A_0.$$

Dynamic cut aggregation

Hybrid aggregation

Corollary

The maximum number of iterations of an L-shaped algorithm with dynamic cut aggregation, where the dynamic partitioning scheme $\mathcal{D}$ satisfies

$$\mathcal{S}^k = \mathcal{S}^N \quad \forall \mathcal{S}^k \in \mathcal{D},\ k > N$$

for some N, is given by

$$N + A(\mathcal{S}^N)\left(\left[1 + A_L(\mathcal{S}^N)(b-1)\right]^m - 1\right)$$

Dynamic cut aggregation - Aggregation schemes

- Dynamic aggregation
  - **SelectUniform**
  - **SelectDecaying**
  - **SelectClosest**
  - **SelectClosestToReference**
- Cluster aggregation
  - **ClusterByReference**
  - **K-medoids**
- Hybrid aggregation

Numerical experiments - SSN

Provision bandwidth in a network before the precise point-to-point demands are known.

Numerical experiments - SSN

Provision bandwidth in a network before the precise point-to-point demands are known.

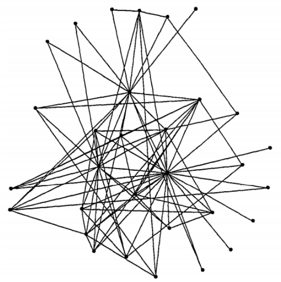


Figure: Network topology in SSN problem [*Sen et al (1994)*]

Numerical experiments - SSN

Provision bandwidth in a network before the precise point-to-point demands are known.

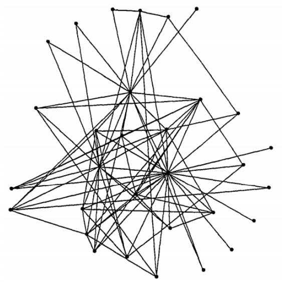


Figure: Network topology in SSN problem [*Sen et al (1994)*]

SAA instance of $n = 10\,000$ scenarios yields a relativevly tight confidence interval around the optimum.

Numerical experiments - Parameter tuning

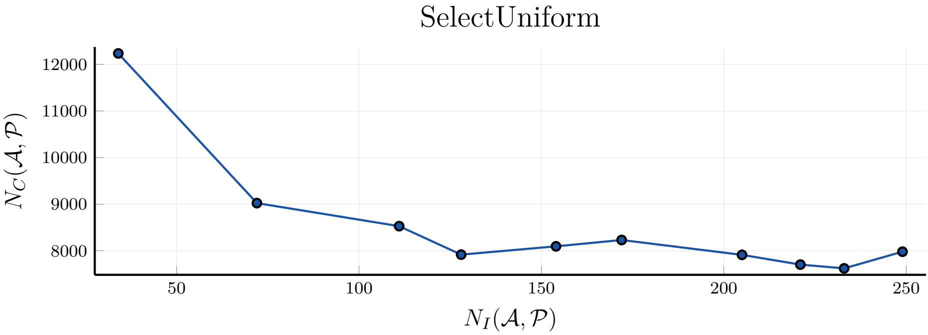


Figure: Empirical complexity for $\mathcal{P} = SSN$ with $n = 1000$ when using the **SelectUniform** decision rule.

Numerical experiments - Parameter tuning

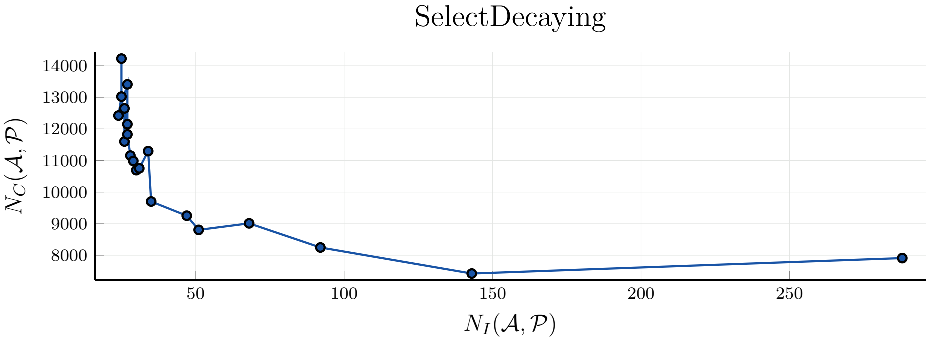


Figure: Empirical complexity for $\mathcal{P} = SSN$ with $n = 1000$ when using the **SelectDecaying** decision rule.

Numerical experiments - Parameter tuning

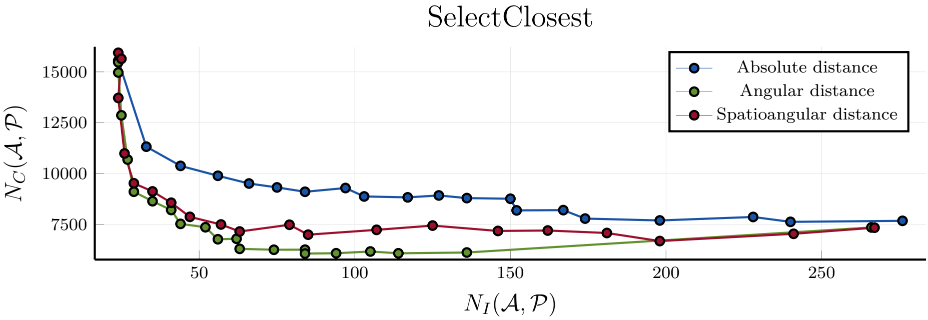


Figure: Empirical complexity for $\mathcal{P} = SSN$ with $n = 1000$ when using the **SelectClosest** decision rule.

Numerical experiments - Parameter tuning

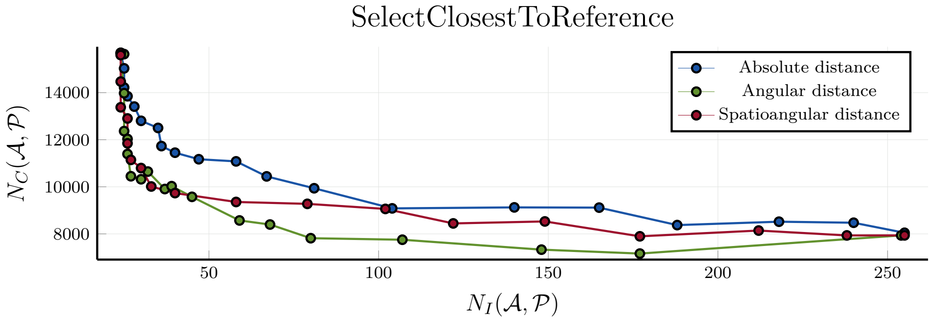


Figure: Empirical complexity for $\mathcal{P} = SSN$ with $n = 1000$ when using the **SelectClosestToReference** decision rule.

Numerical experiments - Parameter tuning

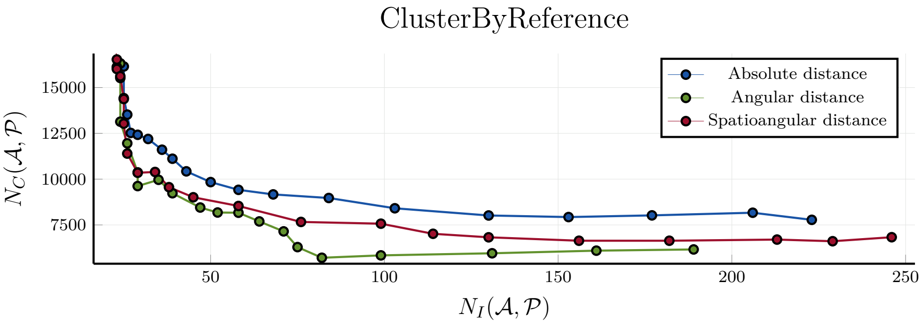


Figure: Empirical complexity for $\mathcal{P} = SSN$ with $n = 1000$ when using the **ClusterByReference** cluster rule.

Numerical experiments - Parameter tuning

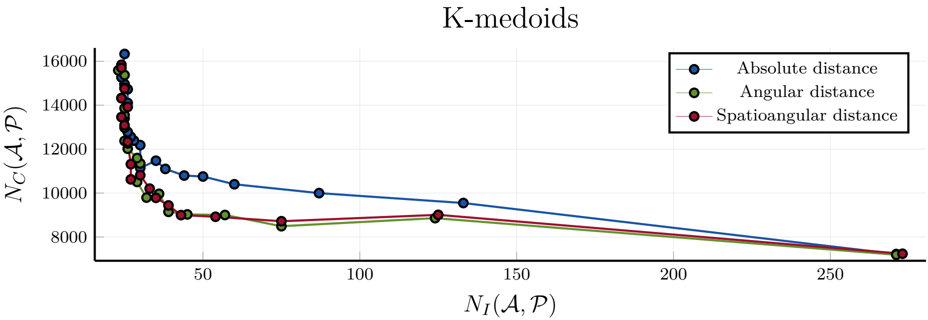


Figure: Empirical complexity for $\mathcal{P} = SSN$ with $n = 1000$ when using **K-medoids** cluster rule.

Numerical experiments - Small-scale SSN

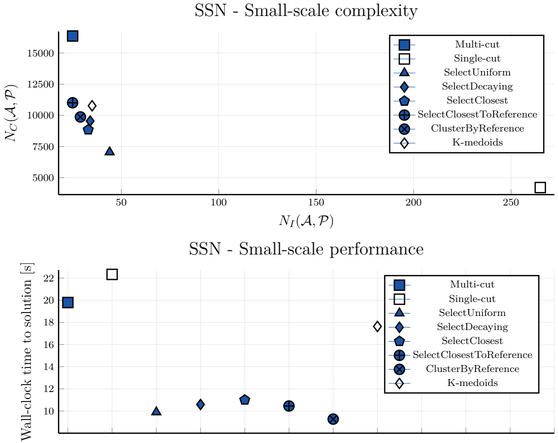


Figure: Empirical complexity and wall-clock time to solution for $\mathcal{P} = SSN$ with $n = 1000$ scenarios.

Numerical experiments - Large-scale SSN

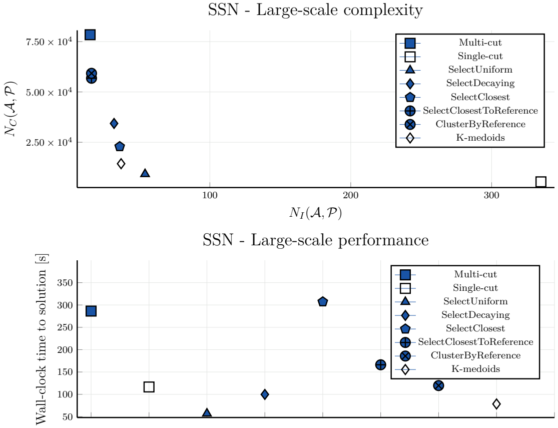


Figure: Empirical complexity and wall-clock time to solution for $\mathcal{P} = SSN$ with $n = 10\,000$ scenarios.

Numerical experiments - Large-scale SSN

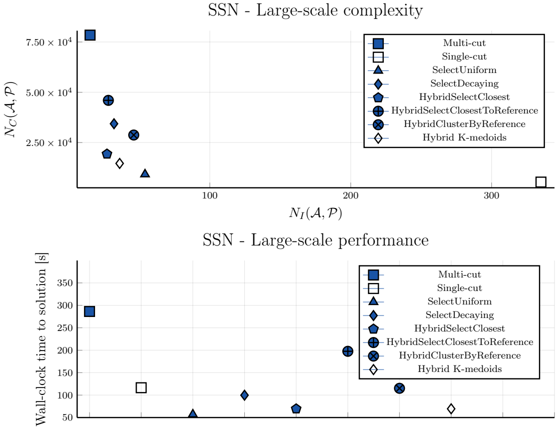


Figure: Empirical complexity and wall-clock time to solution for $\mathcal{P} = SSN$ with $n = 10\,000$ scenarios, using the hybrid fixing strategy.

Numerical experiments - Large-scale day-ahead

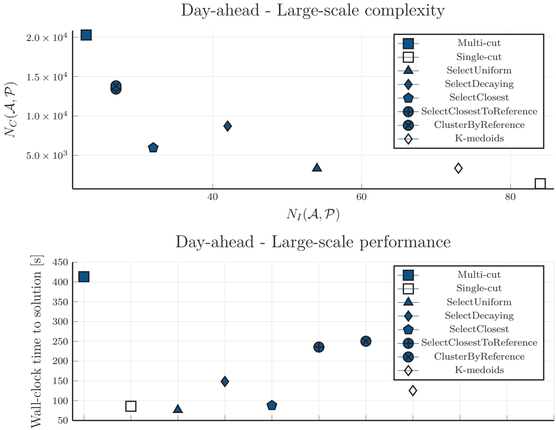


Figure: Empirical complexity and wall-clock time to solution for $\mathcal{P} = DA$ with $n = 1000$ scenarios.

Numerical experiments - Large-scale day-ahead

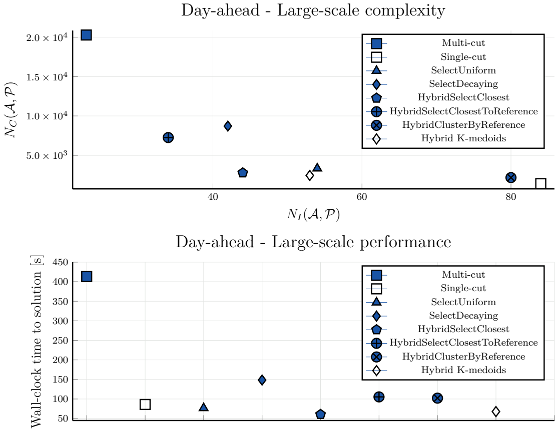


Figure: Empirical complexity and wall-clock time to solution for $\mathcal{P} = DA$ with $n = 1000$ scenarios, using the hybrid fixing strategy.

Cut aggregation - Final Remarks

Discussion

- Worst-case bounds are not reached

Cut aggregation - Final Remarks

Discussion

- Worst-case bounds are not reached
- Hard to reason about average case

Cut aggregation - Final Remarks

Discussion

- Worst-case bounds are not reached
- Hard to reason about average case
- Heuristic aggregation schemes can improve performance

Cut aggregation - Final Remarks

Discussion

- Worst-case bounds are not reached
- Hard to reason about average case
- Heuristic aggregation schemes can improve performance

Summary

- Novel aggregation procedures in L-shaped algorithms

Cut aggregation - Final Remarks

Discussion

- Worst-case bounds are not reached
- Hard to reason about average case
- Heuristic aggregation schemes can improve performance

Summary

- Novel aggregation procedures in L-shaped algorithms
- Convergence is preserved

Cut aggregation - Final Remarks

Discussion

- Worst-case bounds are not reached
- Hard to reason about average case
- Heuristic aggregation schemes can improve performance

Summary

- Novel aggregation procedures in L-shaped algorithms
- Convergence is preserved
- Performance improvements in distributed settings

Cut aggregation - Final Remarks

Discussion

- Worst-case bounds are not reached
- Hard to reason about average case
- Heuristic aggregation schemes can improve performance

Summary

- Novel aggregation procedures in L-shaped algorithms
- Convergence is preserved
- Performance improvements in distributed settings
- Worst-case analysis

Outline

Contribution

- Determine optimal order strategies in a deregulated electricity market

Contribution

- Determine optimal order strategies in a deregulated electricity market
- Complete modeling procedure
  - Data gathering
  - Forecast generation
  - Model formulation
  - Optimization
  - Result visualization

Contribution

- Determine optimal order strategies in a deregulated electricity market
- Complete modeling procedure
  - Data gathering
  - Forecast generation
  - Model formulation
  - Optimization
  - Result visualization

Publications

- Martin Biel. Optimal day-ahead orders using stochastic programming and noise-driven RNNs.
  arXiv preprint arXiv:1910.04510, 2019.
  Submitted for consideration to Energy Systems. Under review

Day-ahead problem - Electricity market

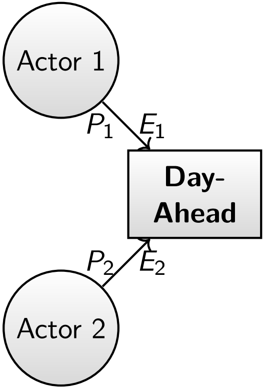
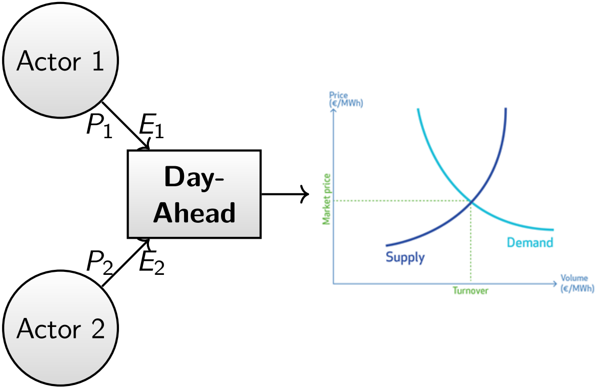


Figure: Deregulated electricity market.

Day-ahead problem - Electricity market



Market closes

Figure: Deregulated electricity market.

Day-ahead problem - Electricity market

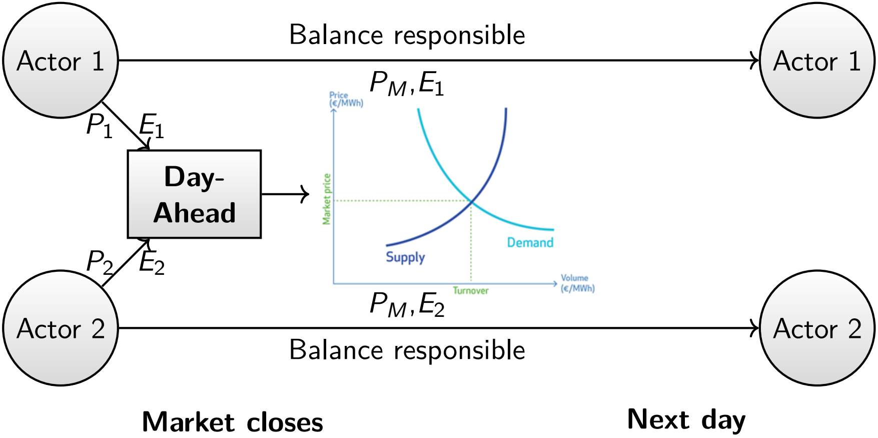


Figure: Deregulated electricity market.

Day-ahead problem - Electricity market

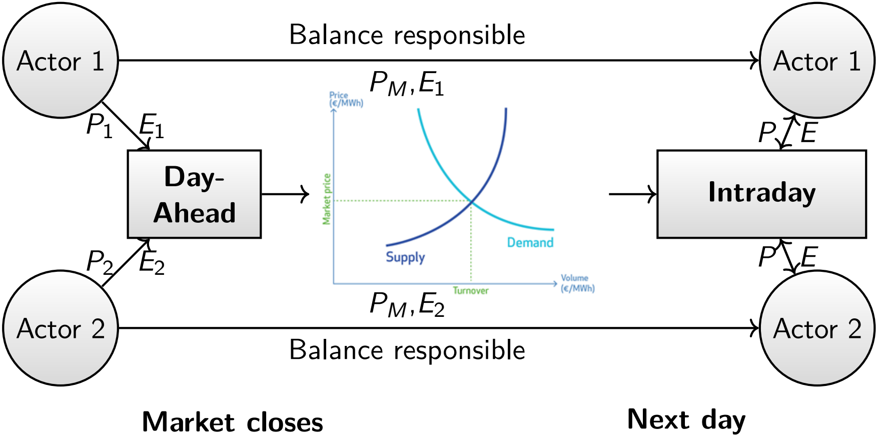


Figure: Deregulated electricity market.

Day-ahead problem - Single order

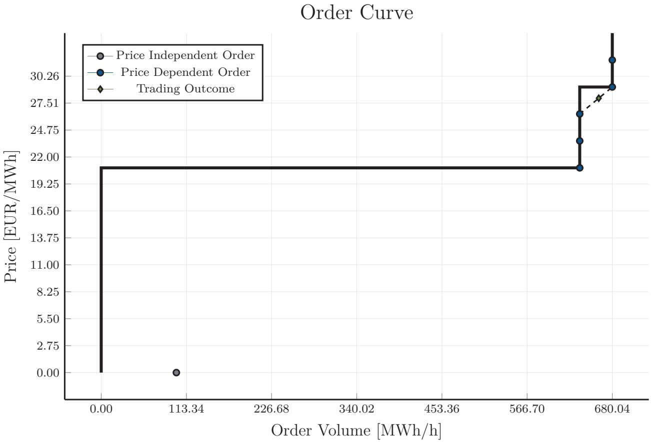


Figure: Single hourly order.

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

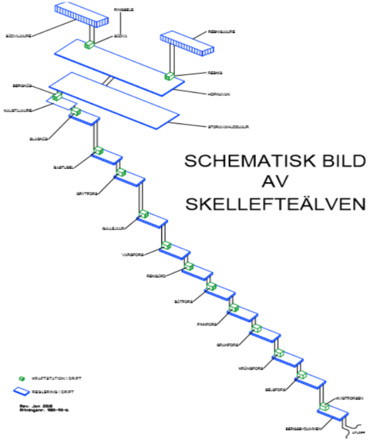


Figure: Schematic of the power stations in Skellefteälven.

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

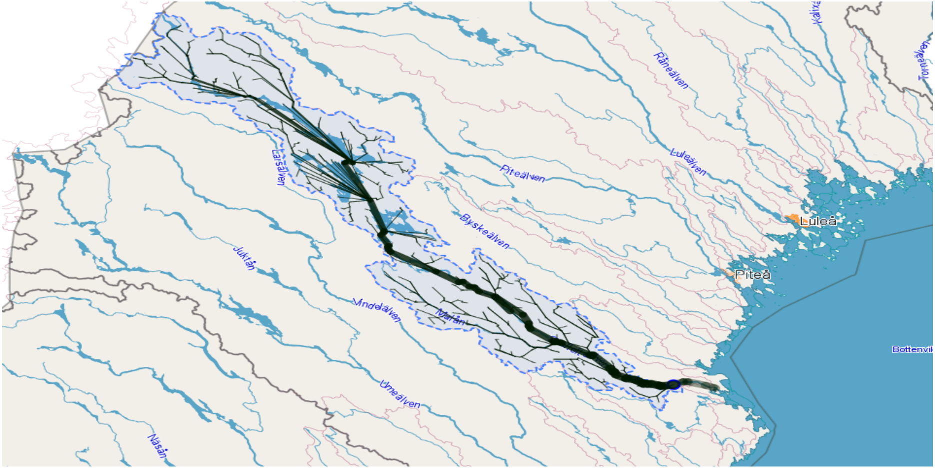


Figure: Mean water flow in Skellefteälven 1999-2018 from SMHI.

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

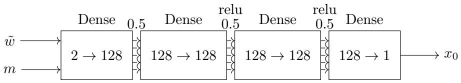


Figure: Initializer network in the price forecaster.

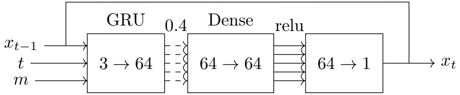


Figure: Sequence generation network in the price forecaster.

Day-ahead problem - Forecasts

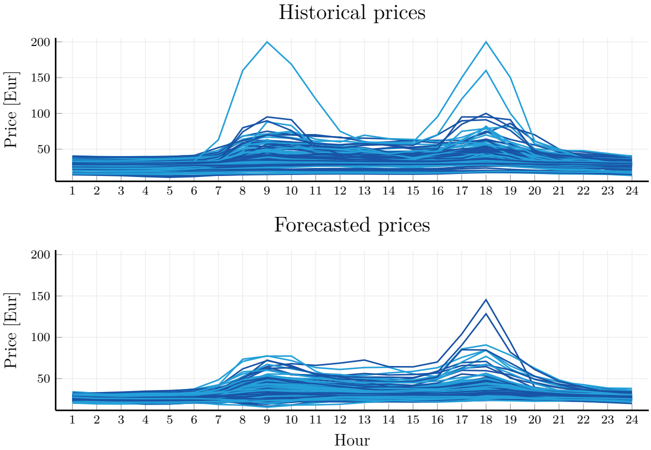


Figure: Historical electricity price curves in January and electricity price curves generated using the RNN forecaster in the same period.

Day-ahead problem - Forecasts

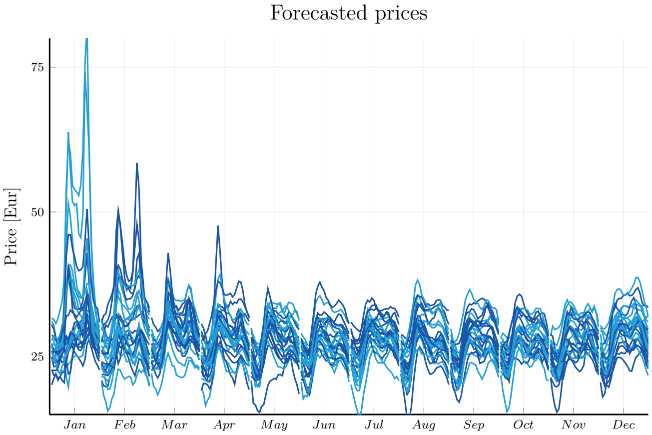


Figure: Daily electricity price curves predicted by the RNN forecaster in every month of the year.

Day-ahead problem - Forecasts

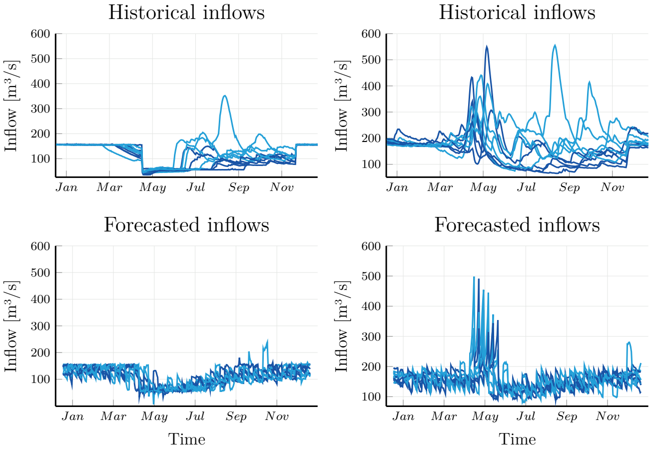


Figure: Historical local inflow in Skellefteälven together and local inflow generated using the RNN forecaster.

Day-ahead problem - Price levels

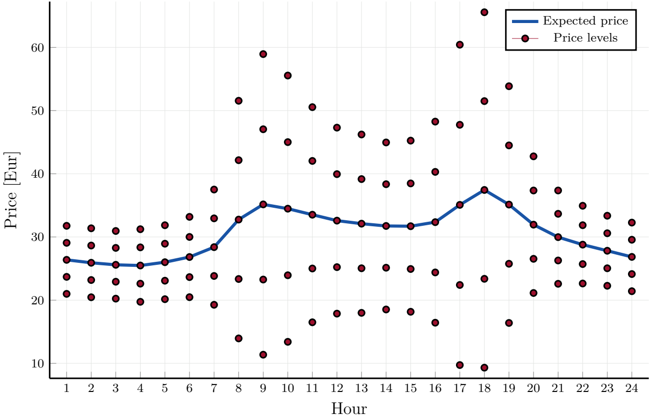


Figure: Expected daily electricity price out of 1000 samples from the RNN forecaster. Two standard deviations above and below the expected price is shown each hour.

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

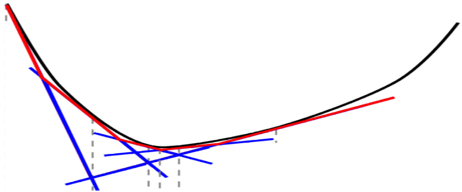


Figure: Polyhedral approximation.

Day-ahead problem - 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 - Results

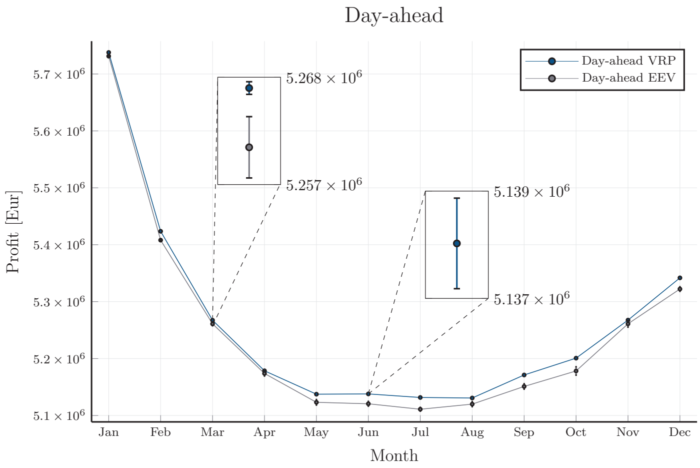


Figure: Seasonal variation of day-ahead VRP and EEV, including 95% confidence intervals.

Day-ahead problem - Results

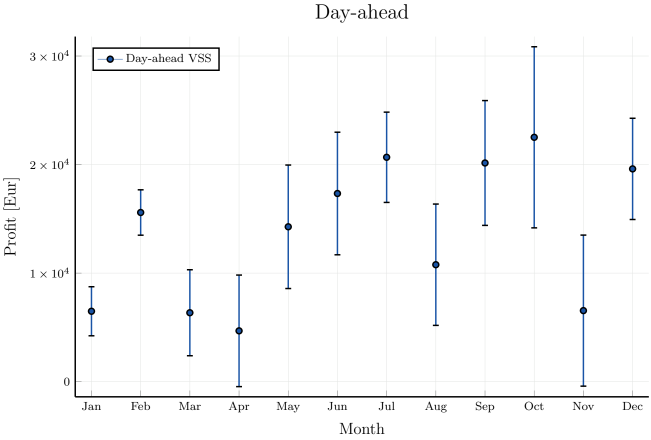


Figure: Seasonal variation of day-ahead VSS, including 90% confidence intervals.

Day-ahead problem - Order strategies

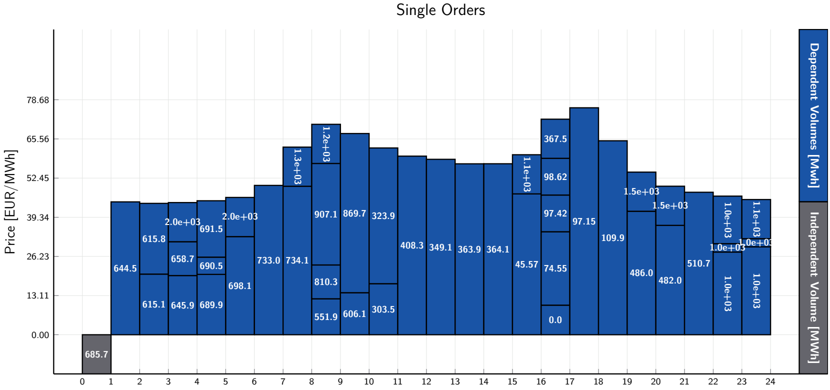


Figure: Day-ahead strategy.

Day-ahead problem - Order strategies

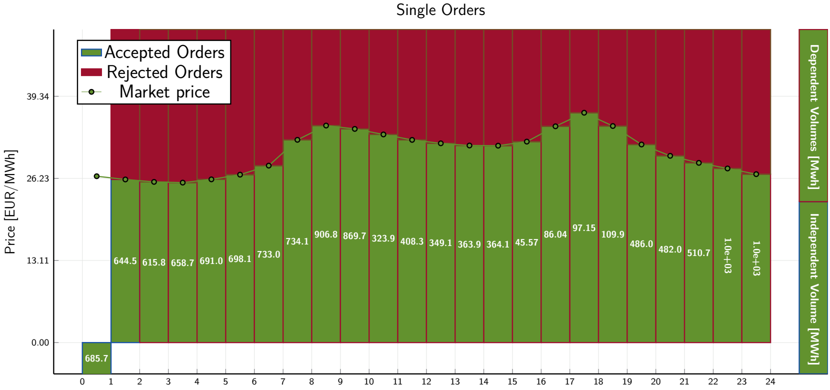


Figure: Result of complete order strategy after realized market price.

Day-ahead problem - Order strategies

Figure: Deterministic order strategy.

Day-ahead problem - Imbalance penalty

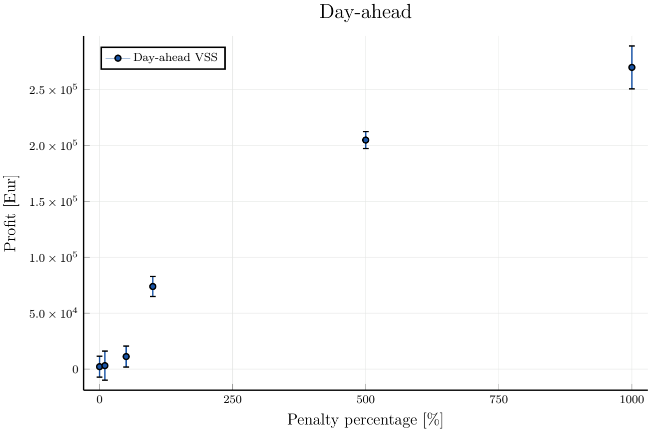


Figure: Day-ahead VSS as a function of imbalance penalty.

Day-ahead problem - Final Remarks

Discussion

- VSS linked to imbalance penalties in the intraday market

Day-ahead problem - Final Remarks

Discussion

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

Day-ahead problem - Final Remarks

Discussion

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

Day-ahead problem - Final Remarks

Discussion

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

Summary

- Large-scale day-ahead problem solved on compute cluster

Day-ahead problem - Final Remarks

Discussion

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

Summary

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

Day-ahead problem - Final Remarks

Discussion

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

Summary

- Large-scale day-ahead problem solved on compute cluster
- Noise-driven recurrent neural networks to sample scenarios
- Tight confidence intervals through sequential SAA

Outline

Conclusion

Summary

- Efficient distributed stochastic programming methods

Conclusion

Summary

- Efficient distributed stochastic programming methods
- Software framework for modeling and solving stochastic programs

Conclusion

Summary

- Efficient distributed stochastic programming methods
- Software framework for modeling and solving stochastic programs
- Performance improvements of the L-shaped algorithm

Conclusion

Summary

- Efficient distributed stochastic programming methods
- Software framework for modeling and solving stochastic programs
- Performance improvements of the L-shaped algorithm
- Effectiveness of the framework illustrated with the day-ahead problem

Conclusion

Summary

- Efficient distributed stochastic programming methods
- Software framework for modeling and solving stochastic programs
- Performance improvements of the L-shaped algorithm
- Effectiveness of the framework illustrated with the day-ahead problem

Outlook on future research

- Multi-stage stochastic programming

Conclusion

Summary

- Efficient distributed stochastic programming methods
- Software framework for modeling and solving stochastic programs
- Performance improvements of the L-shaped algorithm
- Effectiveness of the framework illustrated with the day-ahead problem

Outlook on future research

- Multi-stage stochastic programming
- Sample-based algorithms

Conclusion

Summary

- Efficient distributed stochastic programming methods
- Software framework for modeling and solving stochastic programs
- Performance improvements of the L-shaped algorithm
- Effectiveness of the framework illustrated with the day-ahead problem

Outlook on future research

- Multi-stage stochastic programming
- Sample-based algorithms
- Advanced cut aggregation

Conclusion

Summary

- Efficient distributed stochastic programming methods
- Software framework for modeling and solving stochastic programs
- Performance improvements of the L-shaped algorithm
- Effectiveness of the framework illustrated with the day-ahead problem

Outlook on future research

- Multi-stage stochastic programming
- Sample-based algorithms
- Advanced cut aggregation
- More hydropower decision problems in `StochasticPrograms.jl`