Multi-horizon modelling for 100%-renewable investment planning

Anthony Downward

Joint work with Andy Philpott

Electric Power Optimization Centre
Engineering Science
University of Auckland

INFORMS Annual Meeting, Anaheim, California / Online. October 26, 2021.

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

Outline

Outline

Background

- Multi-horizon stochastic programming
- Implementation and communication of policies

Multi-horizon modelling framework

Medium-term Operational Model

Long-term Planning Model

EMERALD Output

Background

Multi-horizon planning

The concept of multi-horizon modelling is the idea that you are planning for across multiple time-horizons at once (short-term, medium-term, and long-term) and are explicitly accounting for how strategic, tactical and operational decisions influence each other.

Background

Multi-horizon planning

The concept of multi-horizon modelling is the idea that you are planning for across multiple time-horizons at once (short-term, medium-term, and long-term) and are explicitly accounting for how strategic, tactical and operational decisions influence each other.

In this talk, we will be considering this type of problem in the context of stochastic capacity expansion models.

Background

Multi-horizon planning

The concept of multi-horizon modelling is the idea that you are planning for across multiple time-horizons at once (short-term, medium-term, and long-term) and are explicitly accounting for how strategic, tactical and operational decisions influence each other.

In this talk, we will be considering this type of problem in the context of stochastic capacity expansion models.

In the short-term, we have operational decisions that result in immediate costs and revenue; however, at the same time the decision maker is considering capacity expansion decisions that will lead to lower operational costs, or higher revenue in the future.

Background

Implementation and communication of policies

Stochastic programming has been promoted in academia for decades, but has only recently been gaining traction in capacity planning settings within business.

[2] M. Haasnoot, J.H. Kwakkel, W.E. Walker, J. ter Maat, Dynamic adaptive policy pathways: A method for crafting robust decisions for a deeply uncertain world (2013).

Background

Implementation and communication of policies

Stochastic programming has been promoted in academia for decades, but has only recently been gaining traction in capacity planning settings within business.

In part, the delay in acceptance has been due to the limitations in the size of problems that could be modelled, but more significantly the solution to a stochastic program is a policy that adapts to information as it is revealed, and this has been difficult communicate to decision makers.

[2] M. Haasnoot, J.H. Kwakkel, W.E. Walker, J. ter Maat, Dynamic adaptive policy pathways: A method for crafting robust decisions for a deeply uncertain world (2013).

Background

Implementation and communication of policies

Stochastic programming has been promoted in academia for decades, but has only recently been gaining traction in capacity planning settings within business.

In part, the delay in acceptance has been due to the limitations in the size of problems that could be modelled, but more significantly the solution to a stochastic program is a policy that adapts to information as it is revealed, and this has been difficult communicate to decision makers.

The concept of Dynamic Adaptive Pathways has made in-roads in areas where there is deep uncertainty, particularly climate change planning.[2] This, however, is typically more qualitative than quantitative.

[2]M. Haasnoot, J.H. Kwakkel, W.E. Walker, J. ter Maat, Dynamic adaptive policy pathways: A method for crafting robust decisions for a deeply uncertain world (2013).

Outline

Background

- Multi-horizon stochastic programming
- Implementation and communication of policies

Multi-horizon modelling framework

Medium-term Operational Model

Long-term Planning Model

EMERALD Output

Multi-horizon modelling framework

Implementation and communication of policies

In this talk, I will present EMERALD, a multi-horizon electricity capacity planning model built using the JuDGE.jl[3], package for Julia. This package

- allows users to easily implement multi-horizon optimization models using the JuMP modelling language;

[3]A. Downward, R. Baucke, A.B. Philpott, JuDGE.jl: a Julia package for optimizing capacity expansion (2020). (JuDGE stands for Julia Decomposition for Generalized Expansion.)

Multi-horizon modelling framework

Implementation and communication of policies

In this talk, I will present EMERALD, a multi-horizon electricity capacity planning model built using the JuDGE.jl[3], package for Julia. This package

- allows users to easily implement multi-horizon optimization models using the JuMP modelling language;

- applies Dantzig-Wolfe decomposition in order to solve large-scale models;

[3] A. Downward, R. Baucke, A.B. Philpott, JuDGE.jl: a Julia package for optimizing capacity expansion (2020). (JuDGE stands for Julia Decomposition for Generalized Expansion.)

Multi-horizon modelling framework

Implementation and communication of policies

In this talk, I will present EMERALD, a multi-horizon electricity capacity planning model built using the JuDGE.jl[3], package for Julia. This package

- allows users to easily implement multi-horizon optimization models using the JuMP modelling language;
- applies Dantzig-Wolfe decomposition in order to solve large-scale models; and
- applies end-of-horizon risk-measures in objective function and/or the constraints;

[3] A. Downward, R. Baucke, A.B. Philpott, JuDGE.jl: a Julia package for optimizing capacity expansion (2020). (JuDGE stands for Julia Decomposition for Generalized Expansion.)

Multi-horizon modelling framework

Implementation and communication of policies

In this talk, I will present EMERALD, a multi-horizon electricity capacity planning model built using the JuDGE.jl[3], package for Julia. This package

- allows users to easily implement multi-horizon optimization models using the JuMP modelling language;
- applies Dantzig-Wolfe decomposition in order to solve large-scale models; and
- applies end-of-horizon risk-measures in objective function and/or the constraints; and
- outputs an interactive view of the results over the scenario tree, enabling decision makers explore the optimal policy.

[3] A. Downward, R. Baucke, A.B. Philpott, JuDGE.jl: a Julia package for optimizing capacity expansion (2020). (JuDGE stands for Julia Decomposition for Generalized Expansion.)

Multi-horizon modelling framework

What type of problems can be modelled in JuDGE?

JuDGE is a Julia/JuMP-based package that facilitates the modelling of multi-horizon stochastic capacity planning problems.

- $\mathcal{N}$ is the set of nodes in the scenario tree;
- ϕ_n the probability of the state of the world n occurring;
- $\mathcal{P}_n$ the set of nodes on the path to (and including) node n;
- m is the number of expansion variables;
- $z_n \in \mathcal{Z}_+^m$ are the variables for the expansions made at node n;
- y_n is the variable vector for stage-problem n;
- $\mathcal{Y}_n$ is the stage-problem feasibility set.

Multi-horizon modelling framework

What type of problems can be modelled in JuDGE?

JuDGE is a Julia/JuMP-based package that facilitates the modelling of multi-horizon stochastic capacity planning problems.

- $\mathcal{N}$ is the set of nodes in the scenario tree;
- ϕ_n the probability of the state of the world n occurring;
- $\mathcal{P}_n$ the set of nodes on the path to (and including) node n;
- m is the number of expansion variables;
- $z_n \in \mathcal{Z}_+^m$ are the variables for the expansions made at node n;
- y_n is the variable vector for stage-problem n;
- $\mathcal{Y}_n$ is the stage-problem feasibility set.

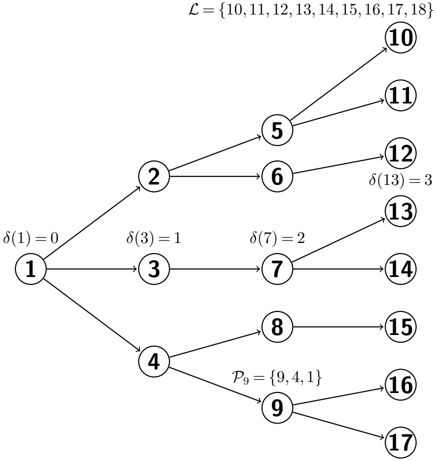
Extensive Form:

$$
\begin{aligned}
\min_{y,z} \quad & \sum_{n\in\mathcal{N}} \phi_n (c_n^\top z_n + q_n^\top y_n) \\
\text{s.t.} \quad & A_n y_n \leq b + D \sum_{h\in\mathcal{P}_n} z_h, && \forall n \in \mathcal{N}, \\
& y_n \in \mathcal{Y}_n, && \forall n \in \mathcal{N}, \\
& z_n \in \mathcal{Z}_+^m, && \forall n \in \mathcal{N}.
\end{aligned}
$$

Multi-horizon modelling framework

What type of problems can be modelled in JuDGE?

JuDGE is a Julia/JuMP-based package that facilitates the modelling of multi-horizon stochastic capacity planning problems.



Extensive Form:

$$
\begin{aligned}
\min_{y,z} \quad & \sum_{n \in \mathcal{N}} \phi_n (c_n^\top z_n + q_n^\top y_n) & \\
\text{s.t.} \quad & A_n y_n \leq b + D \sum_{h \in \mathcal{P}_n} z_h, & \forall n \in \mathcal{N}, \\
& y_n \in \mathcal{Y}_n, & \forall n \in \mathcal{N}, \\
& z_n \in \mathcal{Z}_+^m, & \forall n \in \mathcal{N}.
\end{aligned}
$$

Multi-horizon modelling framework

What type of problems can be modelled in JuDGE?

JuDGE is a Julia/JuMP-based package that facilitates the modelling of multi-horizon stochastic capacity planning problems.

JuDGE applies Dantzig-Wolfe decomposition to the problem by automatically constructing a master problem that handles the investment decisions, and generates columns from the nodal subproblems.

These columns' costs are the operational costs of the nodal subproblems, and the columns' coefficients are the utilized investments.

Extensive Form:

$$
\begin{aligned}
\min_{y,z} \quad & \sum_{n\in\mathcal{N}} \phi_n(c_n^\top z_n + q_n^\top y_n) \\
\text{s.t.} \quad & A_n y_n \leq b + D \sum_{h\in\mathcal{P}_n} z_h, && \forall n \in \mathcal{N}, \\
& y_n \in \mathcal{Y}_n, && \forall n \in \mathcal{N}, \\
& z_n \in \mathcal{Z}_+^m, && \forall n \in \mathcal{N}.
\end{aligned}
$$

Multi-horizon modelling framework

What type of problems can be modelled in JuDGE?

Restricted Master Problem:

$$
\begin{aligned}
\min_{x,w} \quad & \sum_{n\in\mathcal{N}} \phi_n \Big(c_n^\top x_n + \sum_{j\in\mathcal{J}_n} \psi_n^j w_n^j\Big) \\
\text{s.t.} \quad & \sum_{j\in\mathcal{J}_n} \hat{z}_n^j w_n^j \le \sum_{h\in\mathcal{P}_n} x_h, \ \forall n \in \mathcal{N}, \\
& \sum_{j\in\mathcal{J}_n} w_n^j = 1, \qquad \forall n \in \mathcal{N}, \\
& w_n^j, x_n \ge 0, \quad \forall n \in \mathcal{N}, j \in \mathcal{J}_n.
\end{aligned}
$$

The columns are indexed $j \in \mathcal{J}_n$ for each node n, and added to the restricted master problem, with cost ψ_n^j and coefficients $\hat{z}_n^j$.

This problem seeks to choose investments x that minimize the total expected cost, given the columns that have been generated.

Multi-horizon modelling framework

What type of problems can be modelled in JuDGE?

Restricted Master Problem:

$$
\begin{aligned}
\min_{x,w} \quad & \sum_{n\in\mathcal{N}} \phi_n\Big(c_n^\top x_n + \sum_{j\in\mathcal{J}_n} \psi_n^j w_n^j\Big) \\
\text{s.t.} \quad & \sum_{j\in\mathcal{J}_n} \hat{z}_n^j w_n^j \le \sum_{h\in\mathcal{P}_n} x_h, \quad \forall n \in \mathcal{N}, \\
& \sum_{j\in\mathcal{J}_n} w_n^j = 1, \quad \forall n \in \mathcal{N}, \\
& w_n^j, x_n \ge 0, \quad \forall n \in \mathcal{N}, j \in \mathcal{J}_n.
\end{aligned}
$$

The columns are indexed $j \in \mathcal{J}_n$ for each node n, and added to the restricted master problem, with cost ψ_n^j and coefficients $\hat{z}_n^j$.

This problem seeks to choose investments x that minimize the total expected cost, given the columns that have been generated.

Multi-horizon modelling framework

What type of problems can be modelled in JuDGE?

Restricted Master Problem:

$$
\begin{aligned}
\min_{x,w} \quad & \sum_{n\in\mathcal{N}} \phi_n\Big(c_n^\top x_n + \sum_{j\in\mathcal{J}_n} \psi_n^j w_n^j\Big) \\
\text{s.t.} \quad & \sum_{j\in\mathcal{J}_n} \hat{z}_n^j w_n^j \le \sum_{h\in\mathcal{P}_n} x_h, \quad \forall n\in\mathcal{N}, \\
& \sum_{j\in\mathcal{J}_n} w_n^j = 1, \qquad \forall n\in\mathcal{N}, \\
& w_n^j, x_n \ge 0, \quad \forall n\in\mathcal{N}, j\in\mathcal{J}_n.
\end{aligned}
$$

The columns are indexed $j \in \mathcal{J}_n$ for each node n, and added to the restricted master problem, with cost ψ_n^j and coefficients $\hat{z}_n^j$.

This problem seeks to choose investments x that minimize the total expected cost, given the columns that have been generated. (Additional investments cannot decrease the set of feasible columns.)

Multi-horizon modelling framework

What type of problems can be modelled in JuDGE?

Restricted Master Problem:

$$
\begin{aligned}
\min_{x,w} \quad & \sum_{n\in\mathcal{N}} \phi_n \Big(c_n^\top x_n + \sum_{j\in\mathcal{J}_n} \psi_n^j w_n^j \Big) \\
\text{s.t.} \quad & \sum_{j\in\mathcal{J}_n} \hat{z}_n^j w_n^j \le \sum_{h\in\mathcal{P}_n} x_h, \quad \forall n \in \mathcal{N}, \\
& \sum_{j\in\mathcal{J}_n} w_n^j = 1, \quad \forall n \in \mathcal{N}, \\
& w_n^j, x_n \ge 0, \quad \forall n \in \mathcal{N}, j \in \mathcal{J}_n.
\end{aligned}
$$

This problem is solved without any integer variable restrictions, since dual variables are needed for the column generation process.

If the optimal solution is not naturally integer, JuDGE supports both MIP solves for the master, and branch-and-price to find integer feasible solutions.

Multi-horizon modelling framework

What type of problems can be modelled in JuDGE?

This problem is solved without any integer variable restrictions, since dual variables are needed for the column generation process.

If the optimal solution is not naturally integer, JuDGE supports both MIP solves for the master, and branch-and-price to find integer feasible solutions.

Restricted Master Problem:

$$
\begin{aligned}
\min_{x,w} \quad & \sum_{n\in\mathcal{N}} \phi_n \Big(c_n^\top x_n + \sum_{j\in\mathcal{J}_n} \psi_n^j w_n^j \Big) \\
\text{s.t.} \quad & \sum_{j\in\mathcal{J}_n} \hat{z}_n^j w_n^j \leq \sum_{h\in\mathcal{P}_n} x_h, && \forall n \in \mathcal{N}, \\
& \sum_{j\in\mathcal{J}_n} w_n^j = 1, && \forall n \in \mathcal{N}, \\
& w_n^j, x_n \geq 0, && \forall n \in \mathcal{N}, j \in \mathcal{J}_n.
\end{aligned}
$$

Multi-horizon modelling framework

Elements of a JuDGE Model

JuDGE enables the formulation of multistage stochastic capacity management problems leveraging the JuMP mathematical modelling language within Julia.

Multi-horizon modelling framework

Elements of a JuDGE Model

JuDGE enables the formulation of multistage stochastic capacity management problems leveraging the JuMP mathematical modelling language within Julia.

Modelling these problems consists of several elements:

- a tree with corresponding data and probabilities for each node;
- a subproblem defined as a JuMP model for each node in the tree; and
- expansion (and/or shutdown) decisions and costs.

Multi-horizon modelling framework

Elements of a JuDGE Model

JuDGE enables the formulation of multistage stochastic capacity management problems leveraging the JuMP mathematical modelling language within Julia.

Modelling these problems consists of several elements:

- a tree with corresponding data and probabilities for each node;
- a subproblem defined as a JuMP model for each node in the tree; and
- expansion (and/or shutdown) decisions and costs.

Multi-horizon modelling framework

Elements of a JuDGE Model

JuDGE enables the formulation of multistage stochastic capacity management problems leveraging the JuMP mathematical modelling language within Julia.

Modelling these problems consists of several elements:

- a tree with corresponding data and probabilities for each node;
- a subproblem defined as a JuMP model for each node in the tree; and
- expansion (and/or shutdown) decisions and costs.

Multi-horizon modelling framework

Elements of a JuDGE Model

JuDGE enables the formulation of multistage stochastic capacity management problems leveraging the JuMP mathematical modelling language within Julia.

Modelling these problems consists of several elements:

- a tree with corresponding data and probabilities for each node;
- a subproblem defined as a JuMP model for each node in the tree; and
- expansion (and/or shutdown) decisions and costs.

Multi-horizon modelling framework

Elements of a JuDGE Model

JuDGE enables the formulation of multistage stochastic capacity management problems leveraging the JuMP mathematical modelling language within Julia.

Modelling these problems consists of several elements:

- a tree with corresponding data and probabilities for each node;
- a subproblem defined as a JuMP model for each node in the tree; and
- expansion (and/or shutdown) decisions and costs.

Given these elements, JuDGE can automatically form the restricted master problem, and provide the machinery necessary for the iterations of the Dantzig-Wolfe algorithm.

Multi-horizon modelling framework

Elements of a JuDGE Model

JuDGE enables the formulation of multistage stochastic capacity management problems leveraging the JuMP mathematical modelling language within Julia.

Modelling these problems consists of several elements:

- a tree with corresponding data and probabilities for each node;
- a subproblem defined as a JuMP model for each node in the tree; and
- expansion (and/or shutdown) decisions and costs.

Given these elements, JuDGE can automatically form the restricted master problem, and provide the machinery necessary for the iterations of the Dantzig-Wolfe algorithm.

Various solvers can be specified for the different models that are being solved. The LP relaxation of the restricted master problem is typically solved with an interior point method, and the subproblems are solved as mixed-integer programs.

Multi-horizon modelling framework

Elements of a JuDGE Model

JuDGE enables the formulation of multistage stochastic capacity management problems leveraging the JuMP mathematical modelling language within Julia.

Modelling these problems consists of several elements:

- a tree with corresponding data and probabilities for each node;
- a subproblem defined as a JuMP model for each node in the tree; and
- expansion (and/or shutdown) decisions and costs.

Given these elements, JuDGE can automatically form the restricted master problem, and provide the machinery necessary for the iterations of the Dantzig-Wolfe algorithm.

Various solvers can be specified for the different models that are being solved. The LP relaxation of the restricted master problem is typically solved with an interior point method, and the subproblems are solved as mixed-integer programs.

Alternatively, JuDGE can formulate the deterministic equivalent problem directly as a JuMP model (mixed-integer program).

Outline

Medium-term Operational Model

Defining the Subproblems

We will present a electricity generation expansion problem for a competitive electricity market with a complete risk market, and go through the steps necessary to model this.

Medium-term Operational Model

Defining the Subproblems

We will present a electricity generation expansion problem for a competitive electricity market with a complete risk market, and go through the steps necessary to model this.

We have sets:

- technologies $t \in \mathcal{T}$;
- load blocks $b \in \mathcal{B}$; and
- hydrological years $h \in \mathcal{H}$.

Medium-term Operational Model

Defining the Subproblems

We will present a electricity generation expansion problem for a competitive electricity market with a complete risk market, and go through the steps necessary to model this.

We have sets:

- technologies $t \in \mathcal{T}$;
- load blocks $b \in \mathcal{B}$; and
- hydrological years $h \in \mathcal{H}$.

The variables are:

- z_t number of units to build for technology t; and
- g_t^{bh} generation from technology t in load block b, with hydrological year h.

Medium-term Operational Model

Defining the Subproblems

We will present a electricity generation expansion problem for a competitive electricity market with a complete risk market, and go through the steps necessary to model this.

We have sets:

- technologies $t \in \mathcal{T}$;
- load blocks $b \in \mathcal{B}$; and
- hydrological years $h \in \mathcal{H}$.

The variables are:

- z_t number of units to build for technology t; and
- g_t^{bh} generation from technology t in load block b, with hydrological year h.

The parameters are:

- d^b demand in load block b;
- u_t initial capacity of technology t;
- U_t capacity of each new unit of technology t; and
- θ_t^b is the capacity factor for technology t in load block b.

Medium-term Operational Model

Defining the Subproblems

Load balance:

$$\sum_{t \in \mathcal{T}} g_t^{bw} = d^b, \quad \forall b \in \mathcal{B}, w \in \mathcal{W},$$

Generation capacity:

$$0 \leq g_t^{bh} \leq \theta_t^b (u_t + z_t U_t) \quad \forall b \in \mathcal{B}, w \in \mathcal{W}, t \in \mathcal{T},$$

Hydro availability:

$$\sum_{b \in B} g_{\text{hydro}}^{bh} \times \Delta_b \leq \mu^h \quad \forall h \in \mathcal{H},$$

Integer expansions:

$$z_t \in [0, 1, \ldots, Z_t] \quad \forall t \in \mathcal{T}, i \in \{1, \ldots, N\}.$$

Medium-term Operational Model

Defining the Subproblems

Load balance:

$$\sum_{t \in \mathcal{T}} g_t^{bw} = d^b, \quad \forall b \in \mathcal{B}, w \in \mathcal{W},$$

Generation capacity:

$$0 \leq g_t^{bh} \leq \theta_t^b (u_t + z_t U_t) \quad \forall b \in \mathcal{B}, w \in \mathcal{W}, t \in \mathcal{T},$$

Hydro availability:

$$\sum_{b \in B} g_{\text{hydro}}^{bh} \times \Delta_b \leq \mu^h \quad \forall h \in \mathcal{H},$$

Integer expansions:

$$z_t \in [0, 1, \dots, Z_t] \quad \forall t \in \mathcal{T}, i \in \{1, \dots, N\}.$$

Medium-term Operational Model

Defining the Subproblems

Load balance:

$$\sum_{t \in \mathcal{T}} g_t^{bw} = d^b, \quad \forall b \in \mathcal{B}, w \in \mathcal{W},$$

Generation capacity:

$$0 \leq g_t^{bh} \leq \theta_t^b (u_t + z_t U_t) \quad \forall b \in \mathcal{B}, w \in \mathcal{W}, t \in \mathcal{T},$$

Hydro availability:

$$\sum_{b \in B} g_{\text{hydro}}^{bh} \times \Delta_b \leq \mu^h \quad \forall h \in \mathcal{H},$$

Integer expansions:

$$z_t \in [0, 1, \ldots, Z_t] \quad \forall t \in \mathcal{T}, i \in \{1, \ldots, N\}.$$

Medium-term Operational Model

Defining the Subproblems

Load balance:

$$\sum_{t\in\mathcal{T}} g_t^{bw} = d^b, \quad \forall b \in \mathcal{B}, w \in \mathcal{W},$$

Generation capacity:

$$0 \leq g_t^{bh} \leq \theta_t^b(u_t + z_t U_t) \quad \forall b \in \mathcal{B}, w \in \mathcal{W}, t \in \mathcal{T},$$

Hydro availability:

$$\sum_{b\in B} g_{\text{hydro}}^{bh} \times \Delta_b \leq \mu^h \quad \forall h \in \mathcal{H},$$

Integer expansions:

$$z_t \in [0, 1, \ldots, Z_t] \quad \forall t \in \mathcal{T}, i \in \{1, \ldots, N\}.$$

Medium-term Operational Model

Objective Functions

The objective function of the nodal subproblem is to minimize the operational costs of the electricity system:

$$\min \quad \sum_{b\in\mathcal{B}} \Delta_b \sum_{h\in\mathcal{H}} \rho_h \sum_{t\in\mathcal{T}} (c_t + \tau e_t) g_t^{bh},$$

where Δ_b is the number of hours corresponding to load block b;
ρ_w is the probability of hydrological year h;
c_t is the marginal cost of technology t;
e_t gives the emissions factor of technology t; and
τ is the carbon tax.

Medium-term Operational Model

Objective Functions

The objective function of the nodal subproblem is to minimize the operational costs of the electricity system:

$$\min \quad \sum_{b \in \mathcal{B}} \Delta_b \sum_{h \in \mathcal{H}} \rho_h \sum_{t \in \mathcal{T}} (c_t + \tau e_t) g_t^{bh},$$

where Δ_b is the number of hours corresponding to load block b;
ρ_w is the probability of hydrological year h;
c_t is the marginal cost of technology t;
e_t gives the emissions factor of technology t; and
τ is the carbon tax.

Medium-term Operational Model

Objective Functions

The objective function of the nodal subproblem is to minimize the operational costs of the electricity system:

$$\min \quad \sum_{b\in\mathcal{B}} \Delta_b \sum_{h\in\mathcal{H}} \rho_h \sum_{t\in\mathcal{T}} (c_t + \tau e_t) g_t^{bh},$$

where Δ_b is the number of hours corresponding to load block b;
ρ_w is the probability of hydrological year h;
c_t is the marginal cost of technology t;
e_t gives the emissions factor of technology t; and
τ is the carbon tax.

Medium-term Operational Model

Objective Functions

The objective function of the nodal subproblem is to minimize the operational costs of the electricity system:

$$\min \quad \sum_{b \in \mathcal{B}} \Delta_b \sum_{h \in \mathcal{H}} \rho_h \sum_{t \in \mathcal{T}} (c_t + \tau e_t) g_t^{bh},$$

where Δ_b is the number of hours corresponding to load block b;
ρ_w is the probability of hydrological year h;
c_t is the marginal cost of technology t;
e_t gives the emissions factor of technology t; and
τ is the carbon tax.

Medium-term Operational Model

Objective Functions

The objective function of the nodal subproblem is to minimize the operational costs of the electricity system:

$$\min \quad \sum_{b\in\mathcal{B}} \Delta_b \sum_{h\in\mathcal{H}} \rho_h \sum_{t\in\mathcal{T}} (c_t + \tau e_t) g_t^{bh},$$

where Δ_b is the number of hours corresponding to load block b;
ρ_w is the probability of hydrological year h;
c_t is the marginal cost of technology t;
e_t gives the emissions factor of technology t; and
τ is the carbon tax.

Due to the decomposition, we must separately account for the cost of investments over the tree:

$$\min \quad \sum_{n\in\mathcal{N}} \phi_n \sum_{t\in\mathcal{T}} C_t x_t,$$

where ϕ_n is the probability of reaching node n, and C_t is the capital cost (per unit) of technology t, and x_t represents the investment decision for technology t in the master problem.

Medium-term Operational Model

Objective Functions

The objective function of the nodal subproblem is to minimize the operational costs of the electricity system:

$$\min \quad \sum_{b\in\mathcal{B}} \Delta_b \sum_{h\in\mathcal{H}} \rho_h \sum_{t\in\mathcal{T}} (c_t + \tau e_t) g_t^{bh},$$

where Δ_b is the number of hours corresponding to load block b;
ρ_w is the probability of hydrological year h;
c_t is the marginal cost of technology t;
e_t gives the emissions factor of technology t; and
τ is the carbon tax.

Due to the decomposition, we must separately account for the cost of investments over the tree:

$$\min \quad \sum_{n\in\mathcal{N}} \phi_n \sum_{t\in\mathcal{T}} C_t x_t,$$

where ϕ_n is the probability of reaching node n, and C_t is the capital cost (per unit) of technology t, and x_t represents the investment decision for technology t in the master problem.

Medium-term Operational Model

Objective Functions

The objective function of the nodal subproblem is to minimize the operational costs of the electricity system:

$$\min \quad \sum_{b \in \mathcal{B}} \Delta_b \sum_{h \in \mathcal{H}} \rho_h \sum_{t \in \mathcal{T}} (c_t + \tau e_t) g_t^{bh},$$

where Δ_b is the number of hours corresponding to load block b;
ρ_w is the probability of hydrological year h;
c_t is the marginal cost of technology t;
e_t gives the emissions factor of technology t; and
τ is the carbon tax.

Due to the decomposition, we must separately account for the cost of investments over the tree:

$$\min \quad \sum_{n \in \mathcal{N}} \phi_n \sum_{t \in \mathcal{T}} C_t x_t,$$

where ϕ_n is the probability of reaching node n, and C_t is the capital cost (per unit) of technology t, and x_t represents the investment decision for technology t in the master problem.

Medium-term Operational Model

Objective Functions

The objective function of the nodal subproblem is to minimize the operational costs of the electricity system:

$$\min \quad \sum_{b\in\mathcal{B}} \Delta_b \sum_{h\in\mathcal{H}} \rho_h \sum_{t\in\mathcal{T}} (c_t + \tau e_t) g_t^{bh},$$

where Δ_b is the number of hours corresponding to load block b;
ρ_w is the probability of hydrological year h;
c_t is the marginal cost of technology t;
e_t gives the emissions factor of technology t; and
τ is the carbon tax.

Due to the decomposition, we must separately account for the cost of investments over the tree:

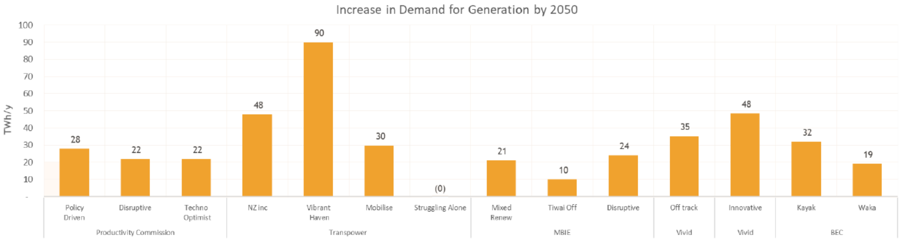
$$\min \quad \sum_{n\in\mathcal{N}} \phi_n \sum_{t\in\mathcal{T}} C_t x_t,$$

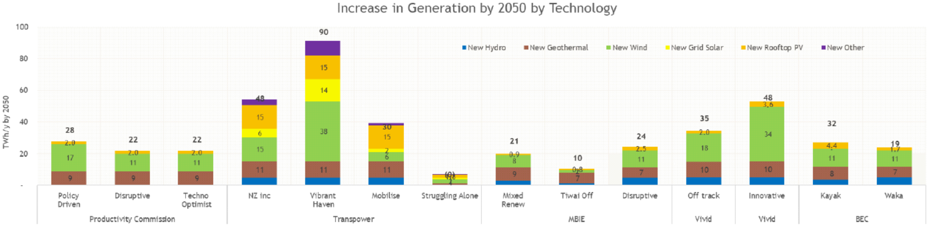
where ϕ_n is the probability of reaching node n, and C_t is the capital cost (per unit) of technology t, and x_t represents the investment decision for technology t in the master problem.

Outline

Scenarios for the future

What investments should be made?





14 scenarios for electricity demand and generation mix in 2050.

There are 14 different ‘optimal’ plans: which should be implemented, if any?

Demand scenarios

The annual demand (TWh) is a stochastic process, modelled using a scenario tree.

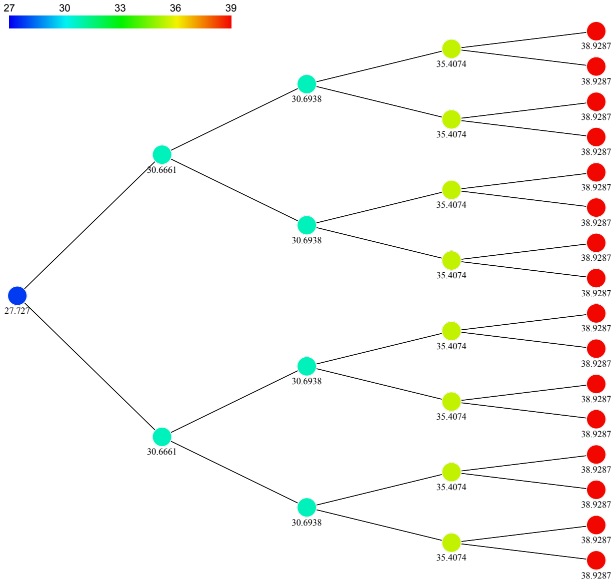
Consumer demand

Industrial demand

Electric Vehicle demand

Tiwai Point Smelter demand

Total demand



Demand scenarios

The annual demand (TWh) is a stochastic process, modelled using a scenario tree.

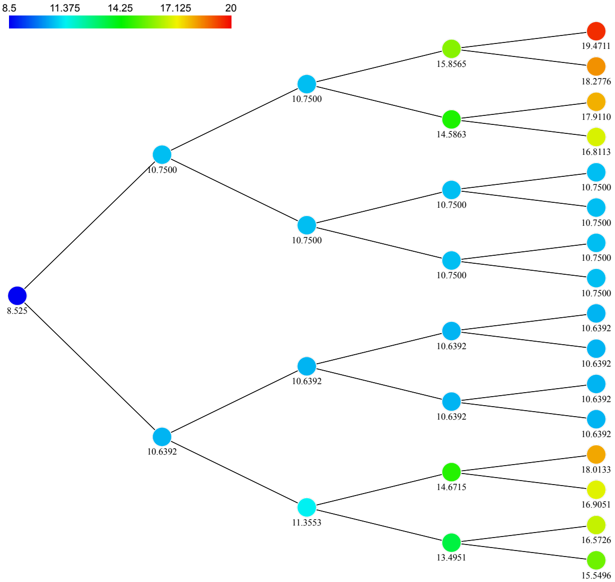
Consumer demand

Industrial demand

Electric Vehicle demand

Tiwai Point Smelter demand

Total demand



Demand scenarios

The annual demand (TWh) is a stochastic process, modelled using a scenario tree.

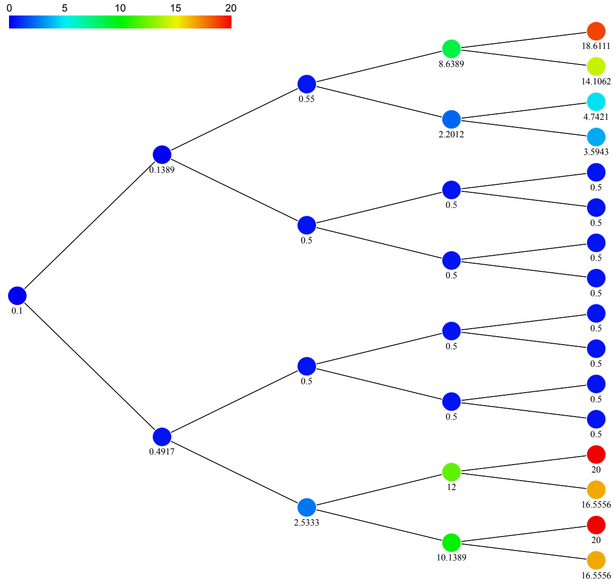
Consumer demand

Industrial demand

Electric Vehicle demand

Tiwai Point Smelter demand

Total demand



Demand scenarios

The annual demand (TWh) is a stochastic process, modelled using a scenario tree.

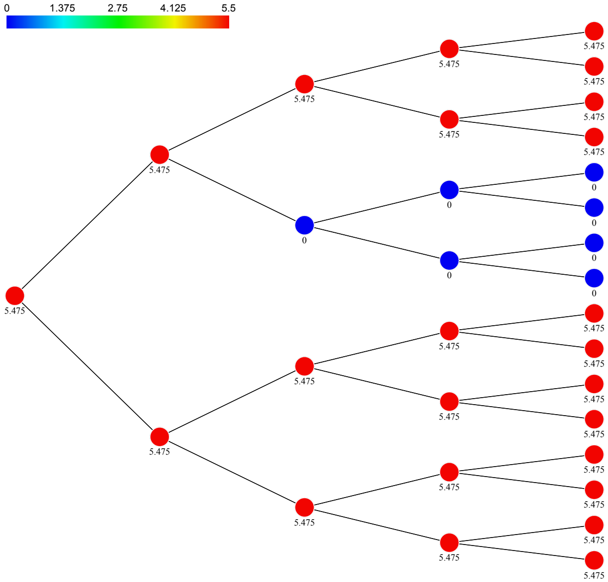
Consumer demand

Industrial demand

Electric Vehicle demand

Tiwai Point Smelter demand

Total demand



Demand scenarios

The annual demand (TWh) is a stochastic process, modelled using a scenario tree.

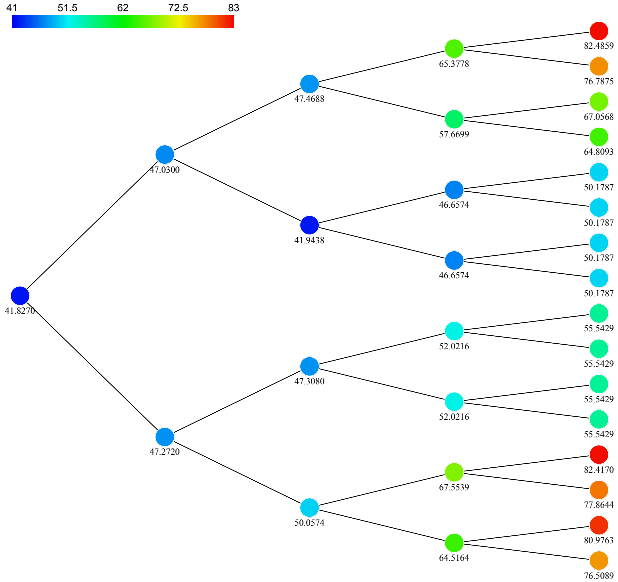
Consumer demand

Industrial demand

Electric Vehicle demand

Tiwai Point Smelter demand

Total demand



Demand scenarios

The annual demand (TWh) is a stochastic process, modelled using a scenario tree.

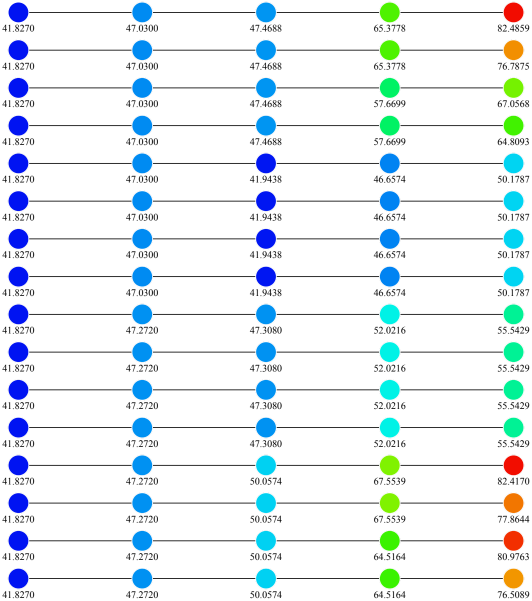
Consumer demand

Industrial demand

Electric Vehicle demand

Tiwai Point Smelter demand

Total demand



Multi-horizon modelling

Creating and solving the JuDGE Model

Once a tree has been created, and a function declared which defines the nodal subproblems, we can create a `JuDGEModel` as follows:

```
model = JuDGEModel(tree, ConditionallyUniformProbabilities,
        sub_problems,optimizer_with_attributes((method=GLPK.INTERIOR)
        -> GLPK.Optimizer(), "msg_lev" => 0, "mip_gap" => 0.0)
        risk = Risk(0.25,0.1))
```

Multi-horizon modelling

Creating and solving the JuDGE Model

Once a tree has been created, and a function declared which defines the nodal subproblems, we can create a JuDGEModel as follows:

```
model = JuDGEModel(tree, ConditionallyUniformProbabilities,
        sub_problems,optimizer_with_attributes((method=GLPK.INTERIOR)
        -> GLPK.Optimizer(), "msg_lev" => 0, "mip_gap" => 0.0)
        risk = Risk(0.25,0.1))
```

Multi-horizon modelling

Creating and solving the JuDGE Model

Once a tree has been created, and a function declared which defines the nodal subproblems, we can create a `JuDGEModel` as follows:

```
model = JuDGEModel(tree, ConditionallyUniformProbabilities,
        sub_problems,optimizer_with_attributes((method=GLPK.INTERIOR)
        -> GLPK.Optimizer(), "msg_lev" => 0, "mip_gap" => 0.0)
        risk = Risk(0.25,0.1))
```

Multi-horizon modelling

Creating and solving the JuDGE Model

Once a tree has been created, and a function declared which defines the nodal subproblems, we can create a JuDGEModel as follows:

```
model = JuDGEModel(tree, ConditionallyUniformProbabilities,
        sub_problems,optimizer_with_attributes((method=GLPK.INTERIOR)
        -> GLPK.Optimizer(), "msg_lev" => 0, "mip_gap" => 0.0)
        risk = Risk(0.25,0.1))
```

Multi-horizon modelling

Creating and solving the JuDGE Model

Once a tree has been created, and a function declared which defines the nodal subproblems, we can create a `JuDGEModel` as follows:

```
model = JuDGEModel(tree, ConditionallyUniformProbabilities,
        sub_problems,optimizer_with_attributes((method=GLPK.INTERIOR)
        -> GLPK.Optimizer(), "msg_lev" => 0, "mip_gap" => 0.0)
        risk = Risk(0.25,0.1))
```

Multi-horizon modelling

Creating and solving the JuDGE Model

Once a tree has been created, and a function declared which defines the nodal subproblems, we can create a `JuDGEModel` as follows:

```
model = JuDGEModel(tree, ConditionallyUniformProbabilities,
        sub_problems,optimizer_with_attributes((method=GLPK.INTERIOR)
        -> GLPK.Optimizer(), "msg_lev" => 0, "mip_gap" => 0.0)
        risk = Risk(0.25,0.1))
```

Multi-horizon modelling

Creating and solving the JuDGE Model

Once a tree has been created, and a function declared which defines the nodal subproblems, we can create a `JuDGEModel` as follows:

```
model = JuDGEModel(tree, ConditionallyUniformProbabilities,
        sub_problems,optimizer_with_attributes((method=GLPK.INTERIOR)
        -> GLPK.Optimizer(), "msg_lev" => 0, "mip_gap" => 0.0)
        risk = Risk(0.25,0.1))
```

If the model passes the in-built testing, ensuring that the JuMP models are set up correctly, the model can be solved using the command:

```
JuDGE.solve(model, Termination = termination(reltol=1e-4)
```

Outline

Communication of JuDGE Solutions

Visualizing the policy

One of the challenges with stochastic multi-horizon optimization is the communication of an optimal policy.

Communication of JuDGE Solutions

Visualizing the policy

One of the challenges with stochastic multi-horizon optimization is the communication of an optimal policy.

JuDGE provides a custom framework to interactively explore the policy, enabling users to understand how the revelation of information influences the investment decisions, but also how these, in turn, affect the operational decisions in the short-term.

Communication of JuDGE Solutions

Visualizing the policy

One of the challenges with stochastic multi-horizon optimization is the communication of an optimal policy.

JuDGE provides a custom framework to interactively explore the policy, enabling users to understand how the revelation of information influences the investment decisions, but also how these, in turn, affect the operational decisions in the short-term.

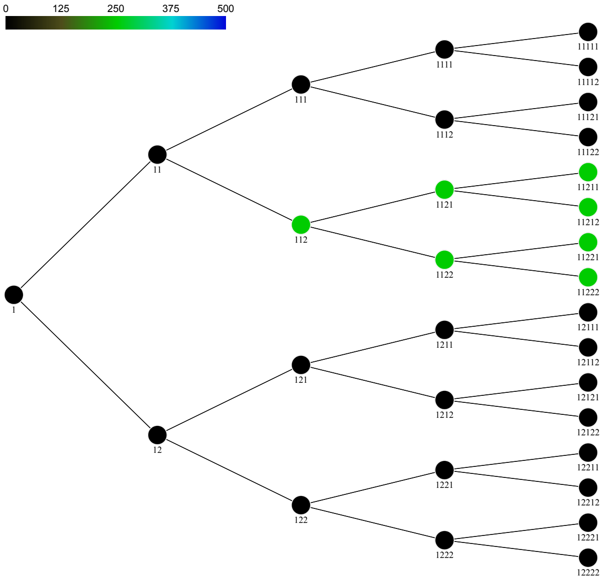
This framework is built around html and javascript, and therefore is very flexible, with the ability to integrate: maps, plots, svg graphics, or any other web-based visualization.

Coal Rankine shutdown

There are two 250MW coal-powered Rankine units near Auckland. These have an O & M cost of $70,000 / MWyr.

EMERALD will time the closure of these plants along with investment in other generation technologies.

Scenario:
No Batteries
No EV smart charging
CO_2 charge $50/$TCO_2$e

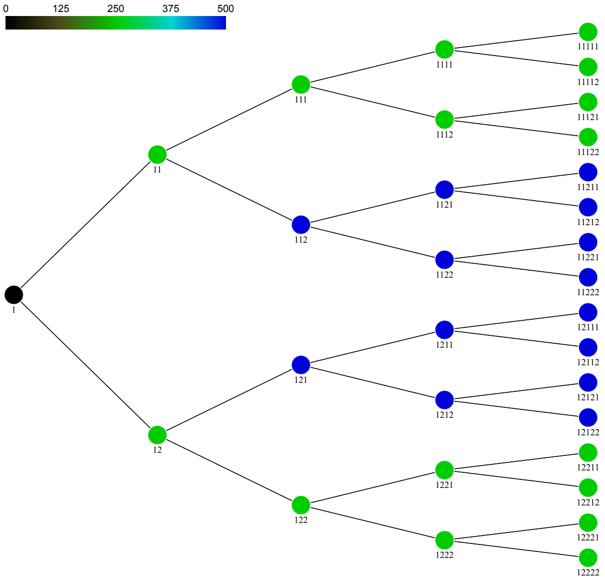


Coal Rankine shutdown

There are two 250MW coal-powered Rankine units near Auckland. These have an O & M cost of $70,000 / MWyr.

EMERALD will time the closure of these plants along with investment in other generation technologies.

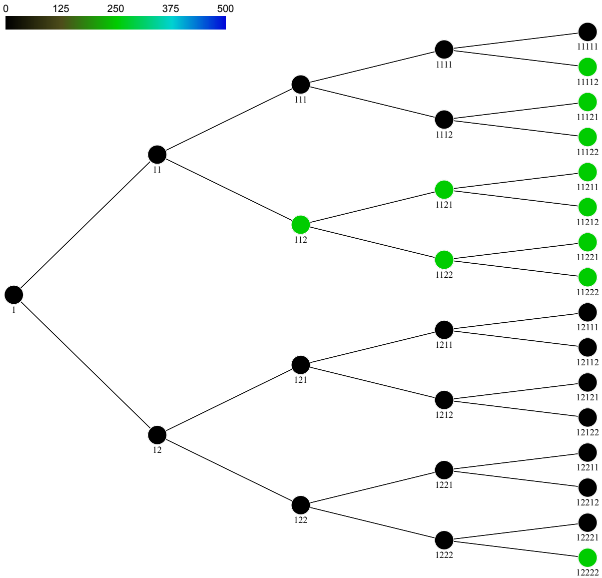
Scenario:
Batteries
No EV smart charging
CO_2 charge $600/$TCO_2$e by 2050



Coal Rankine shutdown

There are two 250MW coal-powered Rankine units near Auckland. These have an O & M cost of \$70,000 / MWyr.

EMERALD will time the closure of these plants along with investment in other generation technologies.



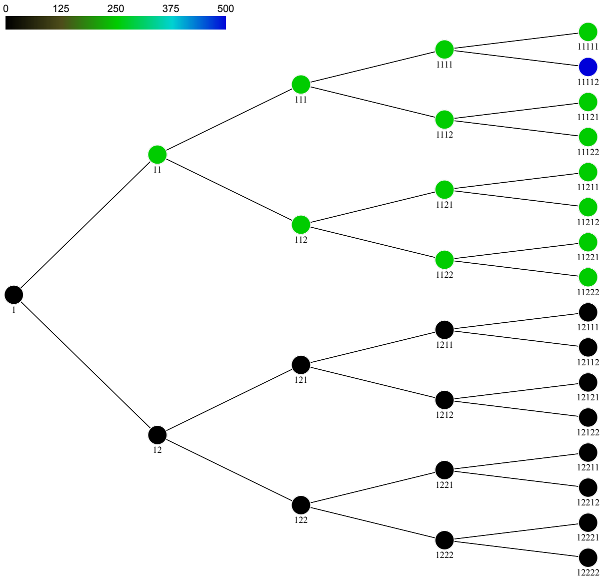
Scenario:
No Batteries
EV smart charging
CO_2 charge \$50/TCO_2e

Coal Rankine shutdown

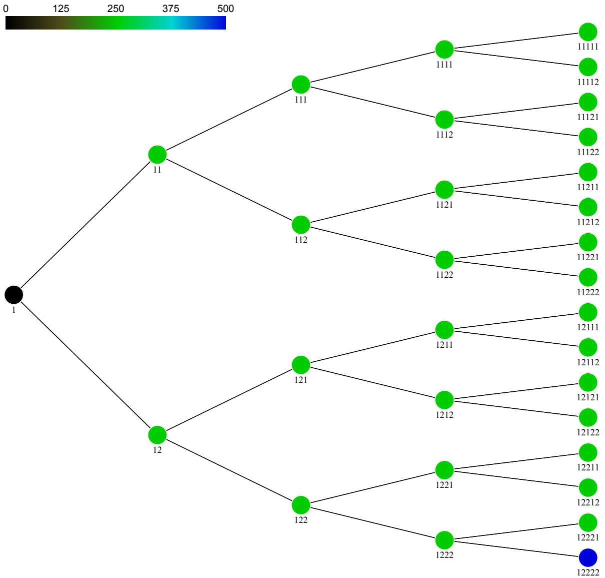
There are two 250MW coal-powered Rankine units near Auckland. These have an O & M cost of $70,000 / MWyr.

EMERALD will time the closure of these plants along with investment in other generation technologies.

Scenario:
Batteries
EV smart charging
CO_2 charge $50/T$CO_2$e



Coal Rankine shutdown



There are two 250MW coal-powered Rankine units near Auckland. These have an O & M cost of $70,000 / MWyr.

EMERALD will time the closure of these plants along with investment in other generation technologies.

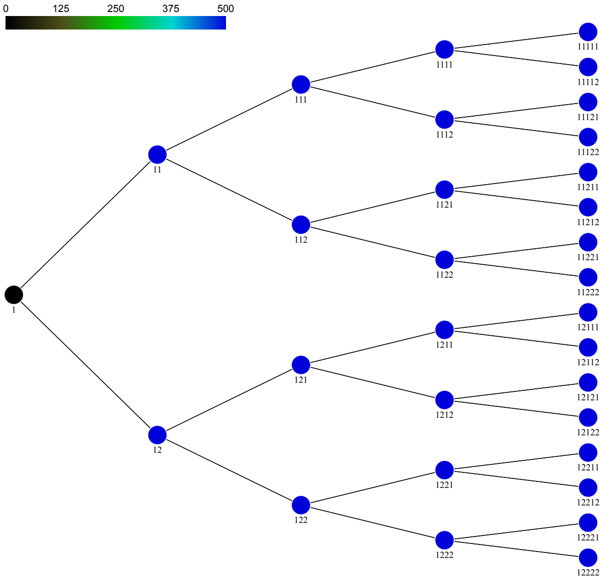
Scenario:
Batteries
EV smart charging
CO_2 charge \$150/TCO_2e by 2050

Coal Rankine shutdown

There are two 250MW coal-powered Rankine units near Auckland. These have an O & M cost of $70,000 / MWyr.

EMERALD will time the closure of these plants along with investment in other generation technologies.

Scenario:
Batteries
EV smart charging
CO_2 charge $600/T$CO_2$e by 2050

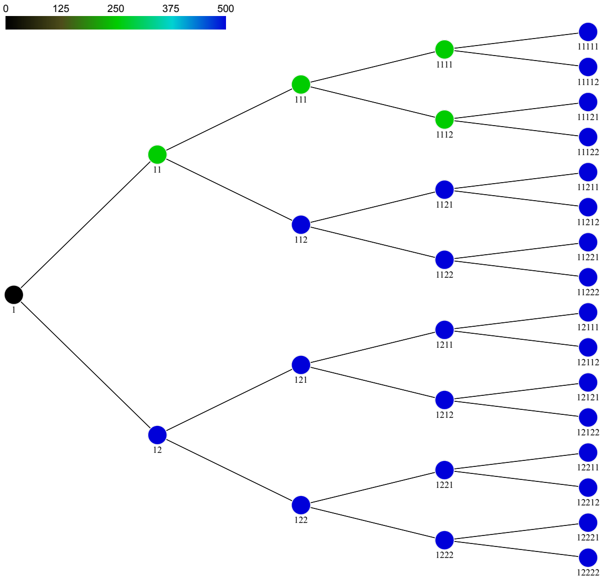


Coal Rankine shutdown

There are two 250MW coal-powered Rankine units near Auckland. These have an O & M cost of $70,000 / MWyr.

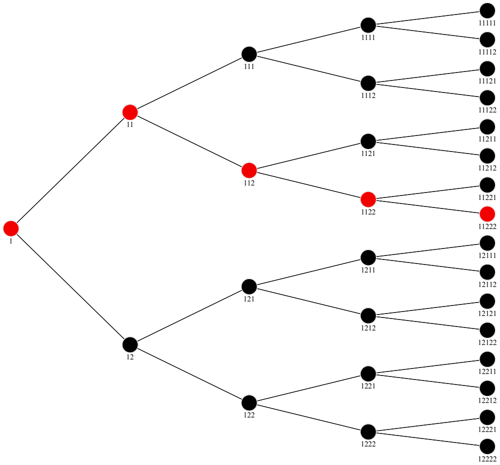
EMERALD will time the closure of these plants along with investment in other generation technologies.

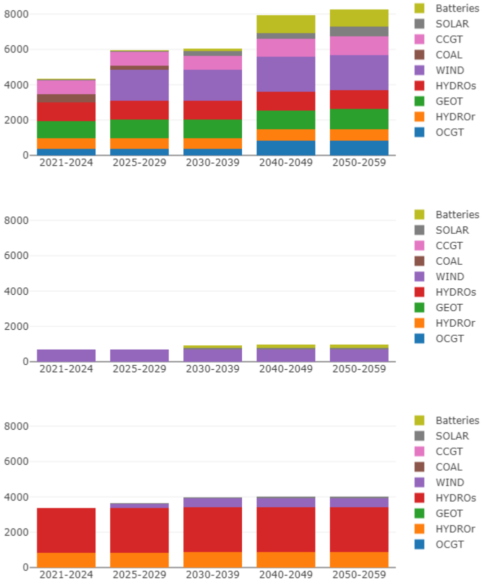
Scenario:
Batteries
EV smart charging
CO_2 charge $600/T$CO_2$e by 2050
Risk averse



Optimal capacity mix

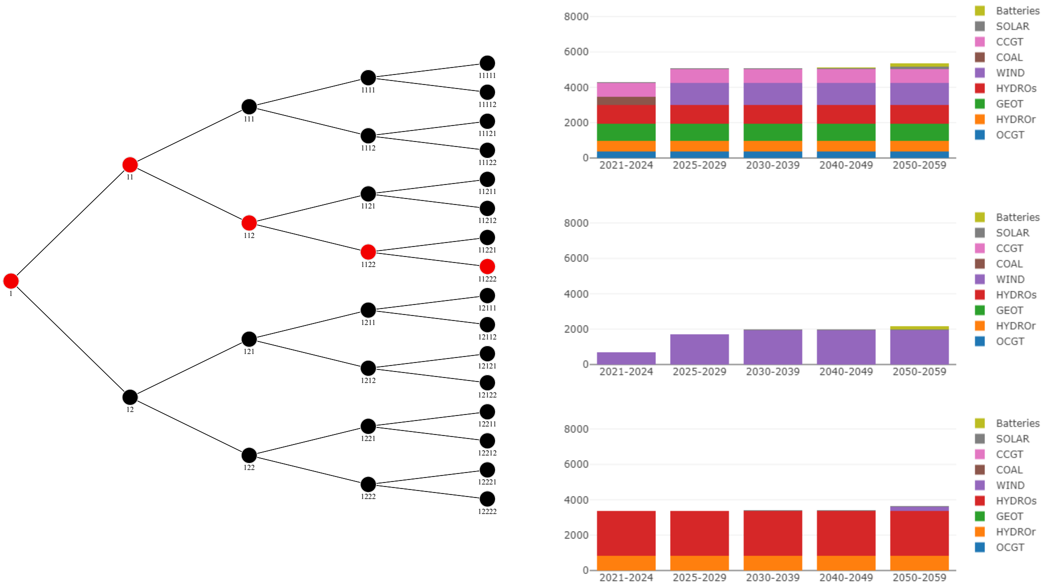
Risk-neutral solution





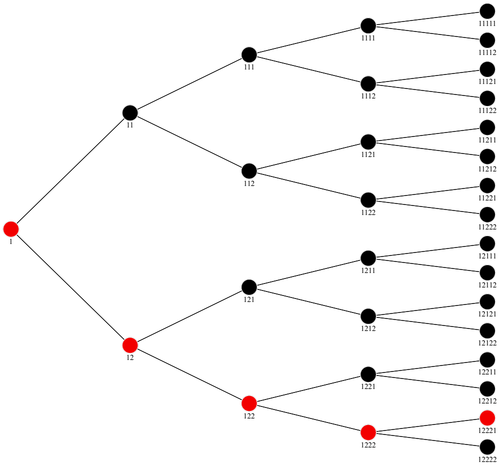
Optimal capacity mix

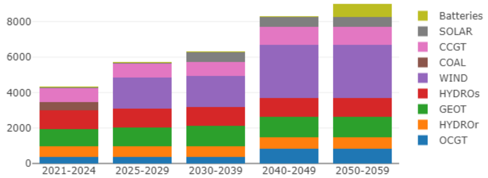
Risk-averse solution

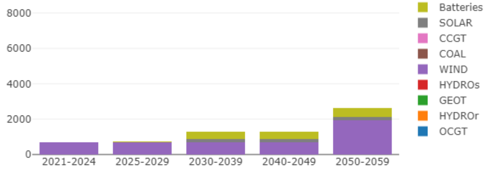


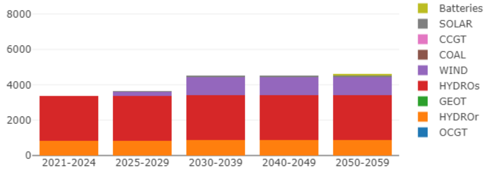
Optimal capacity mix

Risk-neutral solution



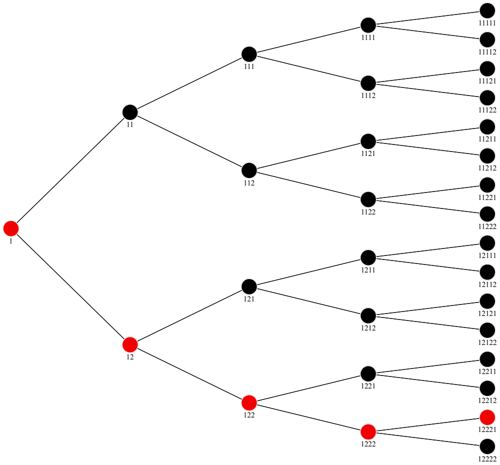


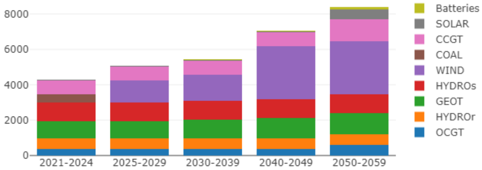


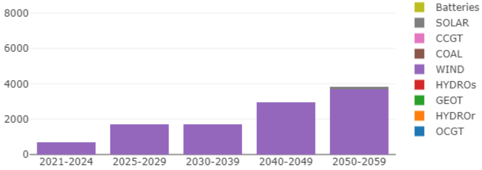


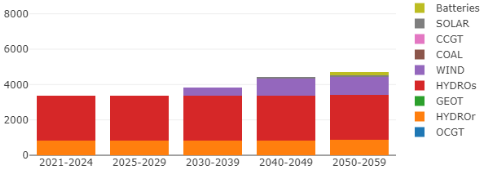
Optimal capacity mix

Risk-averse solution



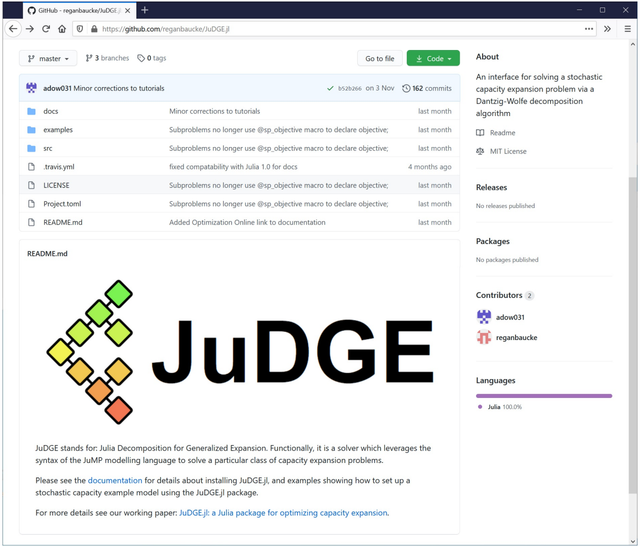






Installing and using JuDGE

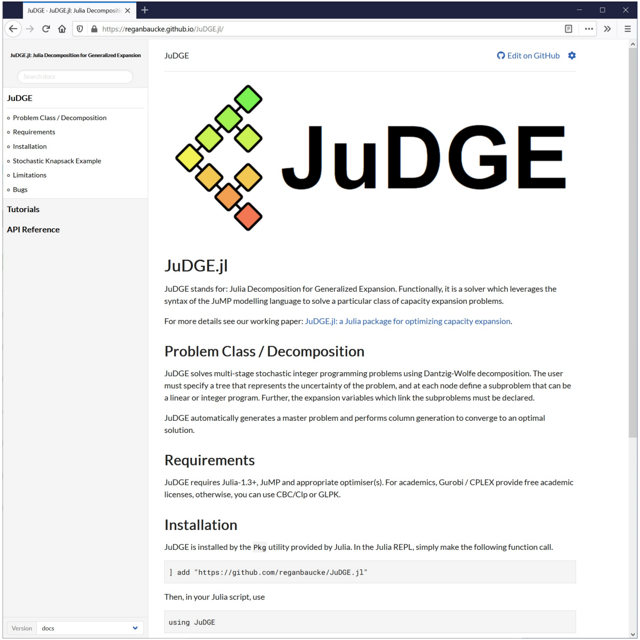
Github Repository



`https://github.com/EPOC-UoA/JuDGE.jl`

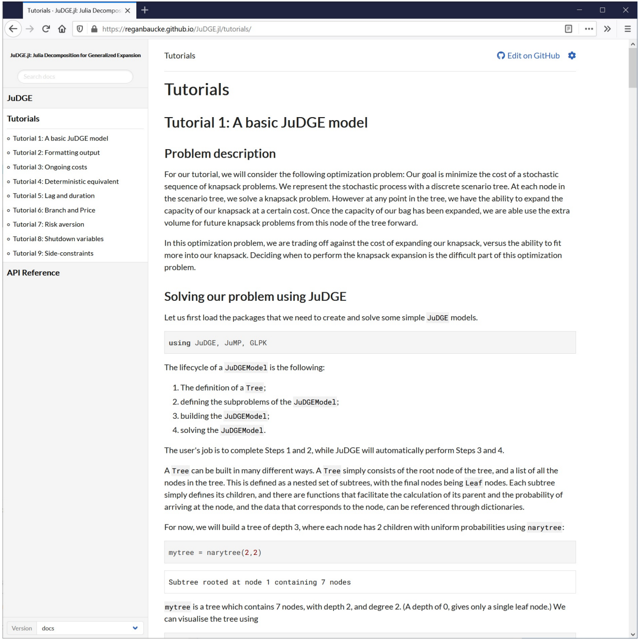
Installing and using JuDGE

Installing the JuDGE Package



Installing and using JuDGE

Tutorials and Examples



Tutorials · JuDGE.jl: Julia Decompos

https://reganbaucke.github.io/JuDGE.jl/tutorials/

JuDGE.jl: Julia Decomposition for Generalized Expansion

Search docs

JuDGE

Tutorials

- Tutorial 1: A basic JuDGE model
- Tutorial 2: Formatting output
- Tutorial 3: Ongoing costs
- Tutorial 4: Deterministic equivalent
- Tutorial 5: Lag and duration
- Tutorial 6: Branch and Price
- Tutorial 7: Risk aversion
- Tutorial 8: Shutdown variables
- Tutorial 9: Side-constraints

API Reference

Version docs

Tutorials

Edit on GitHub

Tutorials

Tutorial 1: A basic JuDGE model

Problem description

For our tutorial, we will consider the following optimization problem: Our goal is minimize the cost of a stochastic sequence of knapsack problems. We represent the stochastic process with a discrete scenario tree. At each node in the scenario tree, we solve a knapsack problem. However at any point in the tree, we have the ability to expand the capacity of our knapsack at a certain cost. Once the capacity of our bag has been expanded, we are able use the extra volume for future knapsack problems from this node of the tree forward.

In this optimization problem, we are trading off against the cost of expanding our knapsack, versus the ability to fit more into our knapsack. Deciding when to perform the knapsack expansion is the difficult part of this optimization problem.

Solving our problem using JuDGE

Let us first load the packages that we need to create and solve some simple `JuDGE` models.

```
using JuDGE, JuMP, GLPK
```

The lifecycle of a `JuDGEModel` is the following:

1. The definition of a `Tree`;
2. defining the subproblems of the `JuDGEModel`;
3. building the `JuDGEModel`;
4. solving the `JuDGEModel`.

The user's job is to complete Steps 1 and 2, while JuDGE will automatically perform Steps 3 and 4.

A `Tree` can be built in many different ways. A `Tree` simply consists of the root node of the tree, and a list of all the nodes in the tree. This is defined as a nested set of subtrees, with the final nodes being `Leaf` nodes. Each subtree simply defines its children, and there are functions that facilitate the calculation of its parent and the probability of arriving at the node, and the data that corresponds to the node, can be referenced through dictionaries.

For now, we will build a tree of depth 3, where each node has 2 children with uniform probabilities using `narytree`:

```
mytree = narytree(2,2)
```

```
Subtree rooted at node 1 containing 7 nodes
```

`mytree` is a tree which contains 7 nodes, with depth 2, and degree 2. (A depth of 0, gives only a single leaf node.) We can visualise the tree using

Thanks for your attention.

Any questions?

`JuDGE.jl` Julia Library `https://github.com/EPOC-UoA/JuDGE.jl`

Contact me: `a.downward@auckland.ac.nz`