SDDP and truck revenue management

Andy Philpott

University of Auckland

August 12, 2022

My background

- Electricity markets.
- Capacity planning for zero-carbon energy systems.
- Optimization of stored hydroelectricity
  - developed doasa model of New Zealand electricity system
  - doasa uses the Stochastic Dual Dynamic Programming (SDDP) algorithm (Pereira and Pinto, 1991).
  - led to SDDP.jl implementation in julia (Dowson and Kapelevich, 2021).
  - SDDP.jl used by New Zealand electricity companies and regulator.
- Recent work with students has applied SDDP.jl to truck revenue management.
  - inspired by problems faced by a NZ startup company.
  - based on previous work on SDDP with Garrett van Ryzin and Michael Frankovich.

Summary

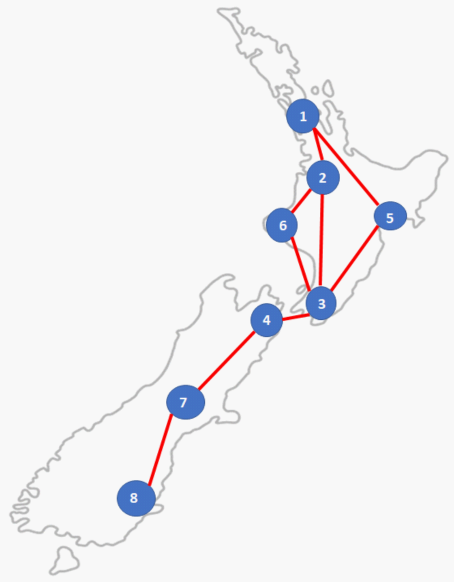
Summary

Network revenue management: known capacity and due date

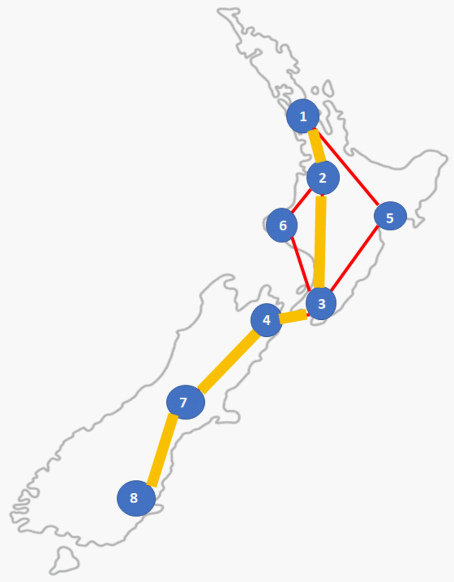
- m network arcs and n shipments arriving over a time horizon of T stages for shipment at end of stage T.
- available truck capacity for each arc at start of stage t denoted $\mathbf{x}(t) = (x_1, x_2, \ldots, x_m)$.
- shipment j uses a_{ij} units of truck capacity on network arcs i, and has a price p_j
- matrix $\mathbf{A} = [a_{ij}]$ with jth column $\mathbf{A}_j$.
- demand for shipment j follows a Poisson process, so in each stage t at most shipment request arrives.
- model as a random revenue vector $\mathbf{P}(t) = (P_1(t), P_2(t), \ldots, P_n(t))$

$$P_j(t) = \begin{cases} p_j, & \text{if a request for shipment } j \text{ occurs,} \\ 0, & \text{otherwise.} \end{cases}$$

Example with three trucks

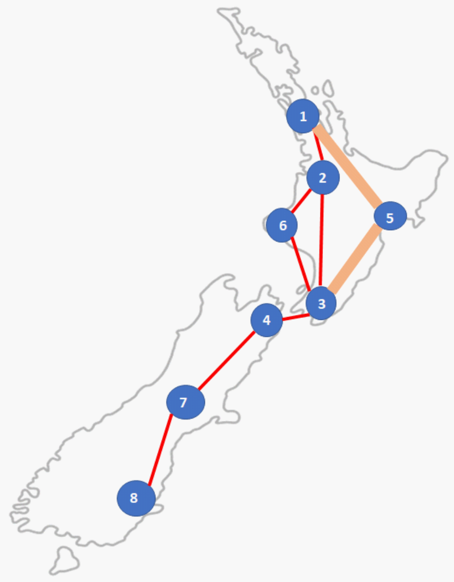


Example with three trucks



Example with three trucks

Example with three trucks



Dynamic programming model

- $R_t(\mathbf{x})$ the maximum expected revenue that can be earned in periods $t, t+1, \ldots, T$, when $\mathbf{x}(t) = \mathbf{x}$ (Bellman function).
- (random) booking action $\mathbf{U} \in \mathcal{U}(\mathbf{x}) = \{0,1\}^n \cap \{\mathbf{u} : \mathbf{Au} \leq \mathbf{x}\}$
- dynamics

$$\mathbf{X}(t+1) = \mathbf{x}(t) - \mathbf{AU}(t)$$

- Bellman function

$$R_t(\mathbf{x}) = \mathbb{E}\Big[\max_{\mathbf{U} \in \mathcal{U}(\mathbf{x})} \{\mathbf{P}(t)^\top \mathbf{u} + R_{t+1}(\mathbf{x} - \mathbf{Au})\}\Big]$$

with

$$R_{T+1}(\mathbf{x}) = 0.$$

Optimal policy

Given $R_{t+1}(\mathbf{x})$, and a realization $(0, 0, \ldots, p_j, 0 \ldots, 0)$ of $\mathbf{P}(t)$ the optimal policy is solution to

$$\max_{u \in \{0,1\}} \{p_j u + R_{t+1}(\mathbf{x} - \mathbf{A}_j u)\}$$

where $\mathbf{A}_j$ is jth column of $\mathbf{A}$.

Solution

$$u_j^*(t, \mathbf{x}, p_j) = \begin{cases} 1, & \textit{if } p_j \geq R_{t+1}(\mathbf{x}) - R_{t+1}(\mathbf{x} - \mathbf{A}_j), \\ 0, & \textit{otherwise.} \end{cases}$$

Random truck capacity

Replace dynamics

$$\mathbf{X}(t+1) = \mathbf{x}(t) - \mathbf{AU}(t)$$

with

$$\mathbf{X}(t+1) = \mathbf{x}(t) - \mathbf{AU}(t) + \delta\mathbf{X}(t)$$

where $\delta\mathbf{X}(t)$ is a random change in truck capacity $\mathbf{x}$ (c.f. random inflow into a hydroelectric reservoir).

Dynamic programming model

- $R_t(\mathbf{x})$ the maximum expected revenue that can be earned in periods $t, t+1, \ldots, T$, when $\mathbf{x}(t) = \mathbf{x}$ (Bellman function).
- $\mathbf{u} \in \mathcal{U}(\mathbf{x} + \delta\mathbf{X})$, where $\mathcal{U}(\mathbf{x}) = \{0,1\}^n \cap \{\mathbf{u} : \mathbf{Au} \leq \mathbf{x}\}$

$$R_t(\mathbf{x}) = \mathbb{E}[\max_{\mathbf{u} \in \mathcal{U}(\mathbf{x}+\delta\mathbf{X})} \{\mathbf{P}(t)^\top \mathbf{u} + R_{t+1}(\mathbf{x} - \mathbf{AU} + \delta\mathbf{X}(t)\}]$$

with

$$R_{T+1}(\mathbf{x}) = 0.$$

Optimal policy

Given $R_{t+1}(\mathbf{x})$, and a realization $(0, 0, \ldots, p_j, 0 \ldots, 0)$ of $\mathbf{P}(t)$,and a realization $\delta\mathbf{x}$ of $\delta\mathbf{X}(t)$ the optimal policy is solution to

$$\max_{u \in \{0,1\}} \{p_j u + R_{t+1}(\mathbf{x} - \mathbf{A}_j u + \delta\mathbf{x})\}$$

where $\mathbf{A}_j$ is jth column of $\mathbf{A}$.

Solution

$$u_j^*(t, \mathbf{x}, p_j, \delta\mathbf{x}) = \begin{cases} 1, & \textit{if } p_j \geq R_{t+1}(\mathbf{x} + \delta\mathbf{x}) - R_{t+1}(\mathbf{x} - \mathbf{A}_j u + \delta\mathbf{x}), \\ 0, & \textit{otherwise.} \end{cases}$$

Different delivery dates (no new trucks)

- Suppose delivery due dates (previously T) are now $d = 1, 2, \ldots$
- For each d, and each $t \leq d$, compute $R_t^d(\mathbf{x})$ the maximum expected revenue earned at end of d when $\mathbf{x}^d(t) = \mathbf{x}$.
- Maintain a state $\mathbf{x}^d(t)$ for each future d.
- Shipping requests u with fixed delivery date d are accepted if $p_j \geq R_{t+1}^d(\mathbf{x}^d) - R_{t+1}^d(\mathbf{x}^d - \mathbf{A}_j u)$
- Can train $R_t^d(\mathbf{x})$ (e.g. in parallel) for each d, and each $t \leq d$.

Flexible delivery dates (no new trucks)

- Suppose delivery due dates for a shipment can be any $d \in D$.
- Shipping requests u with flexible delivery date d are accepted if $p_j \geq \min_{d \in D}\{R^d_{t+1}(\mathbf{x}^d) - R^d_{t+1}(\mathbf{x}^d - \mathbf{A}_j u)\}$
- State $\mathbf{x}^d$ updated to $\mathbf{x}^d - \mathbf{A}_j u$ for best delivery date which is locked in at contract.
- $R^d_t(\mathbf{x})$ can be trained based on this rule. Gives a potentially lower revenue than optimal as no opportunity to rebalance trucks closer to d.

Flexible delivery dates with reallocation (no new trucks)

- Every time period we reallocate accepted shipments to trucks.
- Suppose we receive a request for u with delivery date d, but $p_j < R^d_{t+1}(\mathbf{x}^d) - R^d_{t+1}(\mathbf{x}^d - \mathbf{A}_j u)$
- Previous accepted shipment k that uses resources $\mathbf{A}^d_k$ could shift delivery from day d to d' with $d' \in D$. Let

$$v^{kd'} = \begin{cases} 1, & \text{if shift shipment } k \text{ to } d' \\ 0, & \text{otherwise} \end{cases}.$$

  This gives a change of $\mathbf{x}^d$ to $\mathbf{x}^d + \mathbf{A}^d_k v^{d'} - \mathbf{A}_j u$ and a change of $\mathbf{x}^{d'}$ to $\mathbf{x}^{d'} - \mathbf{A}^d_k v^{d'}$.
- Solve

$$\begin{array}{lll} \max & p_j u + R^d_{t+1}(\mathbf{x}^d + \sum_k \mathbf{A}^d_k \sum_{d' \in D} v^{kd'} - \mathbf{A}_j u) & \\ & \quad + \sum_{d' \in D} R^{d'}_{t+1}(\mathbf{x}^{d'} - \sum_k \mathbf{A}^d_k v^{kd'}) & \\ \text{s.t.} & \sum_{d' \in D} v^{kd'} \leq 1, & k \in K, \\ & u, v^{kd'} \in \{0, 1\}, & k \in K, d' \in D. \end{array}$$

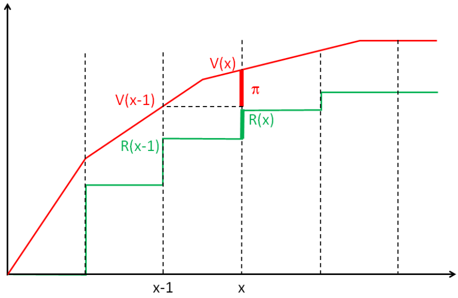
Summary

Continuous approximation

Since $\mathcal{U}(\mathbf{x}) = \{0,1\}^n \cap \{\mathbf{u} : \mathbf{A}\mathbf{u} \leq \mathbf{x}\}$ is a discrete set, $R_t(\mathbf{x})$ may be a discontinuous function. A continuous (outer) approximation $V_t(\mathbf{x}) \geq R_t(\mathbf{x})$ can be obtained by setting

$$\mathcal{U}(\mathbf{x}) = [0,1]^n \cap \{\mathbf{u} : \mathbf{A}\mathbf{u} \leq \mathbf{x}\}$$

which makes $V_t(\mathbf{x})$ continuous and concave. If π is a supergradient to V_t at $\mathbf{x}$ then for any column $\mathbf{A}_j$ of $\mathbf{A}$ we have $V_t(\mathbf{x}) \geq V_t(\mathbf{x} - \mathbf{A}_j) + \pi^\top \mathbf{A}_j$



Bid-price approximation

A bid-pricing policy uses the approximation

$$V_{t+1}(\mathbf{x}+\delta\mathbf{x}) - V_{t+1}(\mathbf{x}-\mathbf{A}_j+\delta\mathbf{x}) \approx \boldsymbol{\pi}^\top \mathbf{A}_j$$

for some vector $\boldsymbol{\pi}$ of bid prices that approximates the supergradient. Then

$$u_j^*(t, \mathbf{x}, p_j, \delta\mathbf{x}) = \begin{cases} 1, & \text{if } p_j \geq \boldsymbol{\pi}^\top \mathbf{A}_j, \\ 0, & \text{otherwise.} \end{cases}$$

Linear programming approximation

$$\begin{array}{lll} \text{DLP}(\mathbf{x},\mathbb{E}[\mathbf{D}(\mathbf{t})]) \text{: max} & \mathbf{p}^\top \mathbf{y} \\ \text{s.t.} & \mathbf{A}\mathbf{y} \leq \mathbf{x} + \mathbb{E}[\mathbf{C}(t)] \\ & 0 \leq \mathbf{y} \leq \mathbb{E}[\mathbf{D}(t)] \end{array}$$

$\mathbf{D}_j(t)$ is expected demand remaining for fare-product j in $t, t+1, \ldots, T$.
$\mathbf{C}_i(t)$ is sum of capacity increments $\delta \mathbf{x}_i(t)$ for leg i over $t, t+1, \ldots, T$.
Optimal dual solution $\boldsymbol{\pi}$ corresponding to $\mathbf{x}$ gives the DLP-bid pricing. So

$$V_{t+1}(\mathbf{x}) - V_{t+1}(\mathbf{x} - \mathbf{A}_j) \approx \boldsymbol{\pi}^\top \mathbf{A}_j.$$

Bid prices are updated by re-solving $DLP(\mathbf{x},\mathbb{E}[\mathbf{D}(\mathbf{t})], \mathbb{E}[\mathbf{C}(\mathbf{t})])$ as booking horizon unfolds.

Supergradients

$$DLP(\mathbf{x},\mathbf{d},\mathbf{c}): \max \quad \mathbf{p}^\top \mathbf{y} \quad \text{s.t.} \quad \mathbf{A}\mathbf{y} \le \mathbf{x}+\mathbf{c}, \quad 0 \le \mathbf{y} \le \mathbf{d}$$

$$DLP^*(\mathbf{x},\mathbf{d},\mathbf{c}): \min \quad \boldsymbol{\pi}^\top(\mathbf{x}+\mathbf{c}) + \boldsymbol{\rho}^\top \mathbf{d} \quad \text{s.t.} \quad \mathbf{A}^\top \boldsymbol{\pi} + \boldsymbol{\rho} \ge \mathbf{p}, \quad \boldsymbol{\pi} \ge \mathbf{0}$$

(π_k, ρ_k) optimal for $DLP^*(x_k, d_k, c_k)$ implies (π_k, ρ_k) feasible for $DLP^*(x, d, c)$ for any (x, d, c) so

$$\begin{aligned} \pi_k^\top (x_k + c_k) + \rho^\top d_k^\top &= V(DLP(x_k, d_k, c_k)) \\ \pi_k^\top (x + c) + \rho_k^\top d &\ge V(DLP(x, d, c)) \end{aligned}$$

Thus (π_k, ρ_k) is a supergradient of $V(DLP(x, d, c))$ at (x_k, d_k, c_k)

$$\begin{aligned} V(DLP(x, d, c)) \le\ & V(DLP(x_k, d_k, c_k)) \\ & + \pi_k^\top (x - x_k) + \rho_k^\top (d - d_k) + \pi_k^\top (c - c_k) \end{aligned}$$

Outer approximation of V(DLP)

Let $\phi(\mathbf{x}, \mathbf{d}, \mathbf{c})$ be the optimal value of $DLP(\mathbf{x}, \mathbf{d}, \mathbf{c})$. Then it is easy to show (e.g. Cooper 2002) that

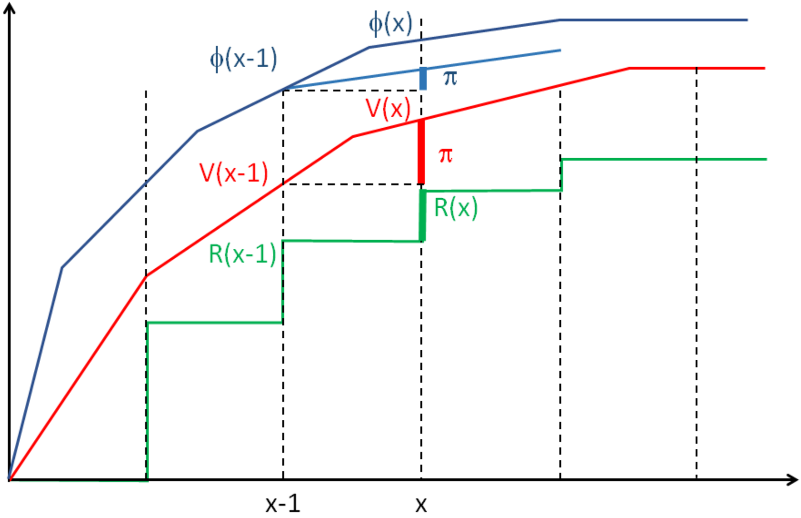
$$V_t(\mathbf{x}) \leq \phi(\mathbf{x}, \mathbb{E}[\mathbf{D}(t)], \mathbb{E}[\mathbf{C}(t)]).$$

Let $\mathbf{x}_k$, $k = 1, \ldots, K$ be resource vectors, $\mathbf{c}_k$, $k = 1, \ldots, K$ be resource increments, $\mathbf{d}_k$, $k = 1, \ldots, K$ be demand vectors, and $(\boldsymbol{\pi}_k, \boldsymbol{\rho}_k)$ the optimal solutions of $DLP^*(\mathbf{x}_k, \mathbf{d}_k, \mathbf{c}_k)$. Then

$$\phi(\mathbf{x}, \mathbf{d}, \mathbf{c}) \leq \min\{\phi(\mathbf{x}_k, \mathbf{d}_k, \mathbf{c}_k) + \boldsymbol{\pi}_k^\top(\mathbf{x} - \mathbf{x_k}) + \boldsymbol{\rho}_k^\top(\mathbf{d} - \mathbf{d_k}) + \boldsymbol{\pi}_k^\top(\mathbf{c} - \mathbf{c_k})\}.$$

The RHS is a polyhedral outer approximation to $\phi_t(\mathbf{x}, \mathbf{d}, \mathbf{c})$, and therefore to $V_t(\mathbf{x})$ when $\mathbb{E}[\mathbf{D}(t)] = \mathbf{d}$ and $\mathbb{E}[\mathbf{C}(t)] = \mathbf{c}$.

Approximations to Bellman function



Approximations to $R(x) - R(x-1)$ from continuous V and DLP ϕ.

Summary

SDDP

- Computing continuous value function $V_t(x)$.
- SDDP (Pereira and Pinto, 1991): Outer approximation of $V_t(x)$ as a polyhedral function

$$\hat{V}_t(x) = \min_{k \in K(t)} \{\alpha_k(t) + \beta_k(t)^\top x\}$$

defined by Kelley cutting planes, i.e.

$$\hat{V}_t(x) = \left\{ \begin{array}{ll} \max & \theta \\ \text{s.t.} & \theta \leq \alpha_k(t) + \beta_k(t)^\top x, \quad k \in K(t). \end{array} \right.$$

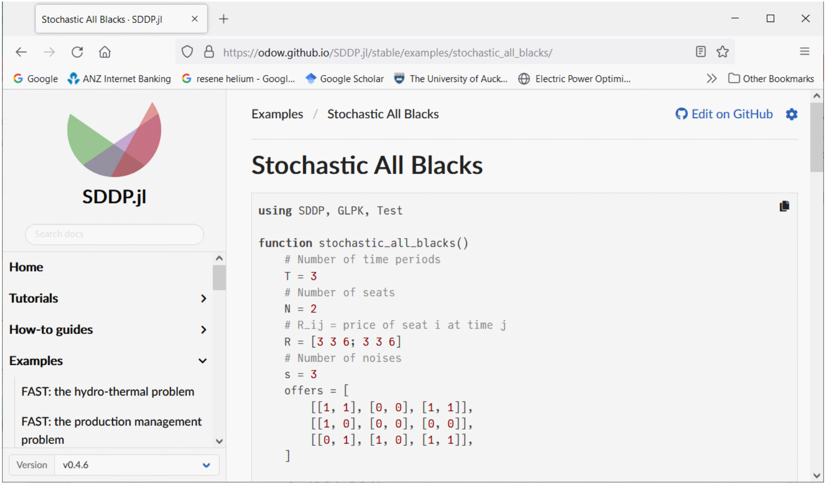
- $\alpha_k(t), \beta_k(t)$ computed iteratively along sample path realizations of random variables.
- Convergence w.p.1 when samples drawn independently (P. and Guan, 2008).

SDDP in Julia

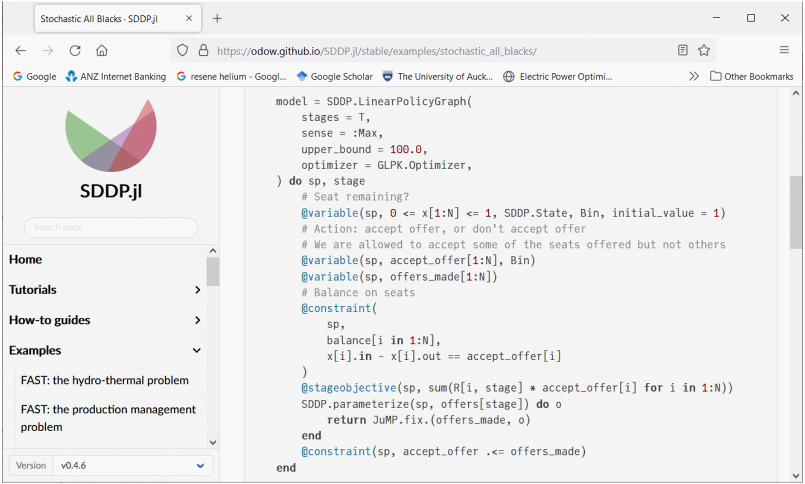
[Dowson & Kapelevich, 2021]

https://odow.github.io/SDDP.jl/stable/

Stochastic All Blacks



Setting up the stage problems



Stochastic All Blacks · SDDP.jl

https://odow.github.io/SDDP.jl/stable/examples/stochastic_all_blacks/

Google | ANZ Internet Banking | resene helium - Googl... | Google Scholar | The University of Auck... | Electric Power Optimi... | Other Bookmarks

SDDP.jl

Search docs

- Home
- Tutorials
- How-to guides
- Examples
  - FAST: the hydro-thermal problem
  - FAST: the production management problem

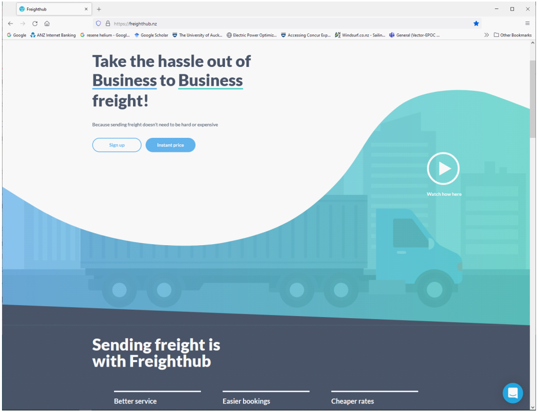
Version v0.4.6

```julia
model = SDDP.LinearPolicyGraph(
    stages = T,
    sense = :Max,
    upper_bound = 100.0,
    optimizer = GLPK.Optimizer,
) do sp, stage
    # Seat remaining?
    @variable(sp, 0 <= x[1:N] <= 1, SDDP.State, Bin, initial_value = 1)
    # Action: accept offer, or don't accept offer
    # We are allowed to accept some of the seats offered but not others
    @variable(sp, accept_offer[1:N], Bin)
    @variable(sp, offers_made[1:N])
    # Balance on seats
    @constraint(
        sp,
        balance[i in 1:N],
        x[i].in - x[i].out == accept_offer[i]
    )
    @stageobjective(sp, sum(R[i, stage] * accept_offer[i] for i in 1:N))
    SDDP.parameterize(sp, offers[stage]) do o
        return JuMP.fix.(offers_made, o)
    end
    @constraint(sp, accept_offer .<= offers_made)
end
```

Training the model

Summary

Freighthub



https://freighthub.nz

The stage problem for trucking

```
do sp, stage

    # state
    @variable(sp, 0.0 <= x[k = 1:arcs, v = 1:vehicles], SDDP.State, initial_value = 0.0)

    # control
    @variable(sp, demand_shipped[i = 1:num_nodes, j = 1:num_nodes, v = 1:vehicles] >= 0.0)
    @variable(sp, arc_shipped[k = 1:arcs, v = 1:vehicles] >= 0.0)
    @variable(sp, demand[i = 1:num_nodes, j = 1:num_nodes])
    @variable(sp, price_matrix[i = 1:num_nodes, j = 1:num_nodes])
    @variable(sp, scenario)
```

The stage problem for trucking

```
# constraints
@constraint(sp, balance[k = 1:arcs, v = 1:vehicles], x[k, v].out == x[k, v].in + arc_shipped[k, v])
@constraint(sp, delivery[i = 1:num_nodes, j = 1:num_nodes], sum(demand_shipped[i, j, v] for v = 1:vehicles) <= demand[i, j])
@constraint(sp, admit[i = 1:num_nodes, j = 1:num_nodes, v = 1:vehicles], demand_shipped[i, j, v] <= admissible[i, j, v] * maxCapacity)
@constraint(sp, arcroute[k = 1:arcs, v = 1:vehicles], arc_shipped[k, v] == sum(demand_shipped[i, j, v] * thevehiclearcs[i, j, v, k]
  for i = 1:num_nodes, j = 1:num_nodes))
# constraint on capacity
@constraint(sp, capacity_limit[k = 1:arcs, v = 1:vehicles], x[k, v].out <= capacity[k, v])
```

```
# parameterize & realise a demand
    SDDP.parameterize(sp, sddp_offers_index) do w

        # fix
        JuMP.fix.(demand, sddp_offers[w])
        JuMP.fix.(price_matrix, sddp_offers_prices[w])
        JuMP.fix.(scenario, w)

    # END PARAMETERISE
    end

    # stage objective
    @stageobjective(sp,
        sum(price_matrix[i, j] * demand_shipped[i, j, v] for i = 1:num_nodes, j = 1:num_nodes, v = 1:vehicles))
```

Demo

Summary

Questions

- SDDP versus Lagrangian relaxation.
- Rate of convergence of SDDP.
- Scalability?
- Is this of any use to Amazon?
- Adoption by Freighthub.

References

- Dowson, O. and Kapelevich, L., 2021. SDDP. jl: a Julia package for stochastic dual dynamic programming. *INFORMS Journal on Computing*, 33(1), pp.27-33.
- Pereira, M.V. and Pinto, L.M., 1991. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1), pp.359-375.
- Philpott, A.B. and Guan, Z., 2008. On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters*, 36(4), pp.450-455.

Literature

1. Improve bid-price approximation by using difference of value function approximations (Bertsimas and Popescu, 2003)
2. Stochastic programming applied to network RM. (de Boer *et al*, Romisch, Higle and Sen, DeMiguel and Mishra, Chen and Homem de Mello)
3. SDDP (Periera and Pinto, 1991) applied in energy planning with some success. (c.f. Powell, Topaloglu and co-authors in vehicle fleet assignment).
4. Re-solving DLP models (Williamson 1992, Cooper, 2002, Talluri and van Ryzin 1998, Maglaras and Meissner, 1986, Reiman and Wang, 2005, Adelman, 2006)