Implementation of the Lagrangian Relaxation Algorithm for Network Revenue Management

Hongzhang "Steve" Shao

May 18, 2025

1 Resources

This project is based on [Topaloglu, 2009]. Both the paper and its dataset are publicly available on Prof. Huseyin Topaloglu's website. You can access the paper directly here. The dataset can be downloaded from this page.

2 The NRM Problem

Consider a **network revenue management problem** with:

- A set of resources (flight legs) $\mathcal{L}$, each with capacity c_i for $i \in \mathcal{L}$.
- A set of products (itineraries) $\mathcal{J}$, each with revenue f_j for $j \in \mathcal{J}$.
  - Each purchase of product j consumes a_{ij} units of capacity from resource i for each i.
- Discrete time horizon $\mathcal{T} = \{1, \dots, \tau\}$.

In each period t:

- **At most one customer arrives**
- The customer requests product j with probability p_{jt}
- $\sum_{j \in \mathcal{J}} p_{jt} \leq 1$

Note that, by adding a dummy itinerary ψ with

$$\begin{aligned} f_\psi &= 0 \\ a_{i\psi} &= 0 && \forall i \in \mathcal{L} \\ p_{\psi t} &= 1 - \sum_{j \in \mathcal{J}} p_{jt} && \forall t \in \mathcal{T} \end{aligned}$$

It can be assumed that in each period t:

- One customer arrives
- The customer requests product j with probability p_{jt}
- $\sum_{j \in \mathcal{J}} p_{jt} = 1$

Let x_{it} be the remaining capacity of resource i at the start of period t. Let $x_t = \big(x_{1t}, x_{2t}, \ldots, x_{|\mathcal{L}|,t}\big)$ be the state vector. Let

$$C = \max_{i \in \mathcal{L}} \; c_i$$
$$\mathcal{C} = \{0, 1, \ldots, C\}$$

and let $\mathcal{C}^{|\mathcal{L}|}$ be the state space.

3 The LR Algorithm

Dynamic Programming Formulation:

- Let $u_{jt} \in \{0,1\}$ indicate whether to accept (1) or reject (0) a request for product j.
- Let $V_t(x_t)$ be the maximum expected revenue from period t to τ given capacities x_t:

$$V_t(x_t) = \max_{u_t \in \mathcal{U}(x_t)} \left\{ \sum_{j \in \mathcal{J}} p_{jt} \left\{ f_j u_{jt} + V_{t+1}\left(x_t - u_{jt} \sum_{i \in \mathcal{L}} a_{ij} e_i \right) \right\} \right\} \tag{DP1}$$

where

$$\mathcal{U}(x_t) = \left\{ u_t \in \{0,1\}^{|\mathcal{J}|} : a_{ij} u_{jt} \le x_{it} \;\; \forall i \in \mathcal{L},\; j \in \mathcal{J} \right\}$$

and e_i is the unit vector with a 1 in the i-th position and 0 elsewhere.

Equivalent Dynamic Program:

- Let $y_{ijt} \in \{0,1\}$ indicate whether to accept (1) or reject (0) **resource** i when a **request for product** j arrives (e.g., it is allowed to partially accept some flight legs when an itinerary uses multiple legs).
- Let ϕ be a **fictitious resource** with infinite capacity.
- Let $y_t = \{y_{ijt} : i \in \mathcal{L} \cup \{\phi\},\; j \in \mathcal{J}\}$.
- Then, $V_t(x_t)$ can be computed as:

$$V_t(x_t) = \max_{y_t \in \mathcal{Y}(x_t)} \left\{ \sum_{j \in \mathcal{J}} p_{jt} \left\{ f_j y_{\phi jt} + V_{t+1}\left(x_t - \sum_{i \in \mathcal{L}} y_{ijt} a_{ij} e_i \right) \right\} \right\} \tag{DP2}$$
$$\text{subject to} \quad y_{ijt} = y_{\phi jt} \quad \forall i \in \mathcal{L},\; j \in \mathcal{J}$$

where

$$\mathcal{Y}_{it}(x_t) = \left\{ y_{it} \in \{0,1\}^{|\mathcal{J}|} : a_{ij} y_{ijt} \le x_{it} \;\; \forall j \in \mathcal{J} \right\} \quad i \in \mathcal{L}$$
$$\mathcal{Y}_{\phi t}(x_t) = \left\{ y_{\phi t} \in \{0,1\}^{|\mathcal{J}|} \right\}$$
$$\mathcal{Y}(x_t) = \mathcal{Y}_{\phi t}(x_t) \prod_{i \in \mathcal{L}} \mathcal{Y}_{it}(x_t) \quad \text{(Cartesian product)}$$

Lagrangian Relaxation:

- Let $\lambda = \{\lambda_{ijt} : i \in \mathcal{L},\ j \in \mathcal{J},\ t \in \mathcal{T}\}$ denote the Lagrangian multiplier.
- The Lagrangian relaxation $V_t^\lambda(x_t)$ is defined as:

$$V_t^\lambda(x_t) = \max_{y_t \in \mathcal{Y}(x_t)} \left\{ \sum_{j \in \mathcal{J}} p_{jt} \left[f_j y_{\phi jt} + \sum_{i \in \mathcal{L}} \lambda_{ijt}(y_{ijt} - y_{\phi jt}) + V_{t+1}^\lambda \left(x_t - \sum_{i \in \mathcal{L}} y_{ijt} a_{ij} e_i \right) \right] \right\} \tag{LR}$$

Lagrangian Relaxation Algorithm:

- **Goal**: The Lagrangian relaxation algorithm aims to find an optimal multiplier λ^* that solves

  $$\min_\lambda V_1^\lambda(c_1)$$

  As shown in [Topaloglu, 2009],

  $$V_t(x_t) \le V_t^\lambda(x_t) \quad \forall x_t \in \mathcal{C}^{|\mathcal{L}|},\ t \in \mathcal{T}$$

  Therefore, $V_1^{\lambda^*}(c_1)$ provides a tight bound to $V_1(c_1)$.

- **Solving $V_1^\lambda(c_1)$ for a given λ**: It has been shown in [Topaloglu, 2009] that $V_1^\lambda(c_1)$ can be solved by concentrating on one resource at a time. Specifically, if $\{\vartheta_{it}^\lambda(x_{it}) : x_{it} \in \mathcal{C}, t \in \mathcal{T}\}$ is a solution to the optimality equation

  $$\vartheta_{it}^\lambda(x_{it}) = \max_{y_{it} \in \mathcal{Y}_{it}(x_{it})} \left\{ \sum_{j \in \mathcal{J}} p_{jt} \left[\lambda_{ijt} y_{ijt} + \vartheta_{i,t+1}^\lambda(x_{it} - a_{ij} y_{ijt}) \right] \right\} \tag{SDP}$$

  for all $i \in \mathcal{L}$, then

  $$V_t^\lambda(x_t) = \sum_{t'=t}^{\tau} \sum_{j \in \mathcal{J}} p_{jt'} \left[f_j - \sum_{i \in \mathcal{L}} \lambda_{ijt'} \right]^+ + \sum_{i \in \mathcal{L}} \vartheta_{it}^\lambda(x_{it}), \tag{1}$$

  where $[z]^+ = \max\{z, 0\}$.

- **Minimizing $V_1^\lambda(c_1)$ over λ**: It has also been shown in [Topaloglu, 2009] that the Lagrangian relaxation $V_1^\lambda(c_1)$ is convex in λ. Thus, the optimal multiplier λ^* can be found by using classical subgradient methods.

Control Policy:

- The control policy is to accept a request for product j at time t if and only if:

  $$f_j \ge \sum_{i \in \mathcal{L}} \sum_{r=1}^{a_{ij}} \left[\vartheta_{i,t+1}^{\lambda^*}(x_{it} - r + 1) - \vartheta_{i,t+1}^{\lambda^*}(x_{it} - r) \right]$$

  That is, a product is accepted if its revenue exceeds the opportunity cost of consumed resources. Specifically, the term $\vartheta_{i,t+1}^{\lambda^*}(x_{it}) - \vartheta_{i,t+1}^{\lambda^*}(x_{it} - 1)$ represents the bid price of resource i at time t.

4 Implementation

Note: [Topaloglu, 2009] did not provide any specific implementation details or the choice of algorithms. Thus, I need to do my own implementation. What follows are my own implementation steps.

4.1 Subroutine: Solving the Single-Resource Dynamic Program

We solve the single-resource optimality equation using tabular backward induction:

Algorithm 1 Subroutine: Tabular Backward Induction for Single-Resource Dynamic Program

Require: Resource i, capacities $\mathcal{C}$, time periods $\mathcal{T}$, probabilities p_{jt}, consumption a_{ij}, multipliers λ_{ijt}

Ensure: Value functions $\vartheta_{it}^{\lambda}(x_{it})$ and optimal decisions $y_{ijt}^{*}(x_{it})$

Initialize $\vartheta_{i,\tau+1}^{\lambda}(x_{i,\tau+1}) \leftarrow 0$ for all $x_{i,\tau+1} \in \mathcal{C}$ ▷ Terminal condition
for $t = \tau$ **down to** 1 **do** ▷ Backward recursion
    for $x_{it} = 0$ **to** C **do** ▷ For each capacity level
        $\vartheta_{it}^{\lambda}(x_{it}) \leftarrow 0$
        $y_{ijt}^{*}(x_{it}) \leftarrow 0$ for all $j \in \mathcal{J}$ ▷ Initialize decision variables
        for $j \in \mathcal{J}$ **do** ▷ For each product
            if $a_{ij} \leq x_{it}$ **then** ▷ Check if capacity is sufficient
                $v_0 \leftarrow \vartheta_{i,t+1}^{\lambda}(x_{it})$ ▷ Value if reject
                $v_1 \leftarrow \lambda_{ijt} + \vartheta_{i,t+1}^{\lambda}(x_{it} - a_{ij})$ ▷ Value if accept
                if $v_1 > v_0$ **then**
                    $y_{ijt}^{*}(x_{it}) \leftarrow 1$ ▷ Accept product j at time t with capacity x_{it}
                end if
            end if
        end for
        $\vartheta_{it}^{\lambda}(x_{it}) \leftarrow \sum_{j \in \mathcal{J}} p_{jt} \Big[\lambda_{ijt}\, y_{ijt}^{*}(x_{it}) + \vartheta_{i,t+1}^{\lambda}\big(x_{it} - a_{ij} y_{ijt}^{*}(x_{it})\big) \Big]$
    end for
end for
return $\{\vartheta_{it}^{\lambda}(x_{it}) : x_{it} \in \mathcal{C}, t \in \mathcal{T}\}$ and $\{y_{ijt}^{*}(x_{it}) : j \in \mathcal{J}, x_{it} \in \mathcal{C}, t \in \mathcal{T}\}$

4.2 Subroutine: Computing the Subgradient of $\vartheta_{it}^{\lambda}(x_{it})$

Consider any resource i. Note that:

$$\begin{aligned}
\vartheta_{it}^{\lambda}(x_{it}) &= \max_{y_{it} \in \mathcal{Y}_{it}(x_{it})} \left\{ \sum_{j \in \mathcal{J}} p_{jt} \left[\lambda_{ijt} y_{ijt} + \vartheta_{i,t+1}^{\lambda}(x_{it} - a_{ij} y_{ijt}) \right] \right\} \\
&= \sum_{j \in \mathcal{J}} p_{jt} \left[\lambda_{ijt} y_{ijt}^{*\lambda} + \vartheta_{i,t+1}^{\lambda}(x_{it} - a_{ij} y_{ijt}^{*\lambda}) \right].
\end{aligned}$$

Let $\mu_{it}(x_{it})$ be the probability mass function of the capacity of resource i at time t. It is given by:

$$\mu_{i,t+1}(x') = \sum_{x} \mu_{it}(x) \sum_{j \in \mathcal{J}} p_{jt} \mathbb{I}\left\{x' = x - a_{ij} y^{*}_{ijt}(x)\right\}.$$

To compute the subgradient of $\vartheta^{\lambda}_{i1}(c_i)$ with respect to λ_{ijt}, let

$$\mu_{i1}(x) = \begin{cases} 1 & \text{if } x = c_i, \\ 0 & \text{otherwise.} \end{cases}$$

Then, we have:

$$\frac{\partial \vartheta^{\lambda}_{i1}(c_i)}{\partial \lambda_{ijt}} = \sum_{x} \mu_{it}(x) y^{*\lambda}_{ijt}(x)$$

Algorithm 2 Forward Pass for Computing $\mu_{it}(x_{it})$

Initialize: $\mu_{i1}(x) \leftarrow 1$ if $x = c_i$, else 0 $\quad \triangleright\ x \in \mathcal{C} = \{0, 1, \ldots, C\}$
for $t = 1$ to $\tau - 1$ **do**
    for each x' in $\mathcal{C}$ **do** $\quad \triangleright\ x' \in \mathcal{C} = \{0, 1, \ldots, C\}$
        $\mu_{i,t+1}(x') \leftarrow 0$
    end for
    for each x in $\mathcal{C}$ **do** $\quad \triangleright\ x \in \mathcal{C} = \{0, 1, \ldots, C\}$
        for each $j \in \mathcal{J}$ **do**
            $x' \leftarrow x - a_{ij} y^{*}_{ijt}(x)$
            if x' is a valid capacity **then** $\quad \triangleright\ x' \in \mathcal{C}$
                $\mu_{i,t+1}(x') \leftarrow \mu_{i,t+1}(x') + \mu_{it}(x) \cdot p_{jt}$
            end if
        end for
    end for
end for

4.3 Subgradient Optimization: Alternative 1

Using (1) and Subsection 4.2, we have:

$$\frac{\partial V^{\lambda}_{1}(c_1)}{\partial \lambda_{ijt}} = \frac{\partial \vartheta^{\lambda}_{i1}(c_i)}{\partial \lambda_{ijt}} - p_{jt} \mathbf{1}\left\{f_j - \sum_{k \in \mathcal{L}} \lambda_{kjt} \geq 0\right\}$$

Note that as a Lagrangian multiplier, $\lambda_{ijt} \geq 0$. Thus, here we use a projected subgradient descent algorithm to optimize λ.

Algorithm 3 Projected Subgradient Descent for Lagrangian Multiplier Optimization

Require: Initial multipliers λ^0, step size α_0, tolerance ϵ, maximum iterations K
Ensure: Optimized multipliers λ^*

$k \leftarrow 0$
$V_{\text{prev}} \leftarrow \infty$
while $k < K$ **do**
  Compute $V_1^{\lambda^k}(c_1)$
  if $|V_1^{\lambda^k}(c_1) - V_{\text{prev}}| < \epsilon$ **then**
    break
  end if
  $V_{\text{prev}} \leftarrow V_1^{\lambda^k}(c_1)$
  Compute subgradient g^k of $V_1^{\lambda^k}(c_1)$ with respect to λ.
  $\alpha_k \leftarrow \frac{\alpha_0}{\sqrt{k+1}}$
  $\lambda^{k+1} \leftarrow \max\{0, \lambda^k - \alpha_k g^k\}$ ▷ Project onto feasible set
  $k \leftarrow k + 1$
end while
return λ^k

4.4 Subgradient Optimization: Alternative 2

Without computing subgradient analytically, we can use the following algorithm to optimize λ:

Algorithm 4 Subgradient Optimization for Lagrangian Relaxation

Require: Initial multiplier λ^0, initial step size α_0, tolerance ϵ, max iterations K
Ensure: Optimal multiplier λ^*

$k \leftarrow 0$
$V_{\text{prev}} \leftarrow \infty$
while $k < K$ **do**
  Compute $V_1^{\lambda^k}(c_1)$ ▷ Evaluate current objective
  if $|V_1^{\lambda^k}(c_1) - V_{\text{prev}}| < \epsilon$ **then**
    break ▷ Convergence achieved
  end if
  $V_{\text{prev}} \leftarrow V_1^{\lambda^k}(c_1)$
  Generate random unit vector d
  Compute $V_1^{\lambda^k+\delta d}(c_1)$ ▷ Perturbed objective
  $g^k \leftarrow \frac{V_1^{\lambda^k+\delta d}(c_1) - V_1^{\lambda^k}(c_1)}{\delta} \cdot d$ ▷ Approximate subgradient
  $\alpha_k \leftarrow \frac{\alpha_0}{\sqrt{k+1}}$ ▷ Update step size
  $\lambda^{k+1} \leftarrow \max\{0, \lambda^k - \alpha_k g^k\}$ ▷ Component-wise projection onto $\Lambda = \{\lambda \geq 0\}$
  $k \leftarrow k + 1$
end while
return λ^k

References

[Topaloglu, 2009] Topaloglu, H. (2009). Using lagrangian relaxation to compute capacity-dependent bid prices in network revenue management. *Operations Research*, 57(3):637–649.