# A Concise Overview of the Lagrangian Relaxation Algorithm for Network Revenue Management

Hongzhang "Steve" Shao

May 15, 2025

## 1 Problem Formulation

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 $\mathcal{T} = \{1, \ldots, \tau\}$.

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 $\psi$ with

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

we can assume 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 $x_t = \left(x_{1t}, x_{2t}, \ldots, x_{|\mathcal{L}|,t}\right)$ be the state vector. Let

$$
\begin{aligned}
C &= \max_{i \in \mathcal{L}} \ c_i \\
\mathcal{C} &= \{0, 1, \ldots, C\}
\end{aligned}
$$

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

## 2 Preliminaries

**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 $\tau$ 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

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

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

An **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., we allow partially accepting some flight legs when an itinerary uses multiple legs).
- Let $\phi$ 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

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

The **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:

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

# 3 Pseudocode

The Lagrangian relaxation algorithm aims to find an optimal multiplier $\lambda^*$ 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 $V_1(c_1)$.

**Subgradient Optimization**: Computing $V_1^{\lambda}(c_1)$ is costly, and we cannot find its gradient analytically. However, the Lagrangian relaxation $V_1^{\lambda}(c_1)$ is convex in $\lambda$, as shown in [Topaloglu, 2009]. Thus, we can use subgradient methods to find the optimal multiplier $\lambda^*$. We use a Classical Projected Subgradient Descent algorithm as a simple, robust solution.

**Algorithm 1** Subgradient Optimization for Lagrangian Relaxation

**Require:** Initial multiplier $\lambda^0$, initial step size $\alpha_0$, tolerance $\epsilon$, max iterations $K$
**Ensure:** Optimal multiplier $\lambda^*$

$k \leftarrow 0$
$V_{\text{prev}} \leftarrow \infty$
**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 $\Lambda = \{\lambda \ge 0\}$
  $k \leftarrow k+1$
**end while**
**return** $\lambda^k$

**Solving $V_1^{\lambda}(c_1)$ for a given $\lambda$**: 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

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

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

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

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

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

**Algorithm 2** 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 $\lambda_{ijt}$

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

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

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

That is, we accept a product if its revenue exceeds the opportunity cost of consumed resources. 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 $t$.

# 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.