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

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 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

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

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

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

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],

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

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:

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

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 The subgradient of $V_1^\lambda(\mathbf{c}_1)$

Computing the subgradient is important for updating λ_{ijt} when minimizing $V_1^\lambda(c_1)$. However, [Topaloglu, 2009] does not explicitly provide the subgradient of $V_1^\lambda(c_1)$. In this section, we show how to compute the subgradient of $V_1^\lambda(c_1)$ with respect to a specific Lagrange multiplier λ_{ijt}.

Single-Resource Value Function and Optimal Policy: Recall that the value function $\vartheta_{it}^\lambda(x_{it})$ for resource i with capacity x_{it} at time t is given by the Bellman equation:

$$\vartheta_{it}^\lambda(x_{it}) = \max_{y_{ijt}\in\{0,1\}} \left\{ \sum_{j\in\mathcal{J}} p_{jt} \left[\lambda_{ijt} y_{ijt} + \vartheta_{i,t+1}^\lambda(x_{it} - a_{ij} y_{ijt}) \right] \right\}$$

Let $y_{ijt}^{*\lambda}(x_{it})$ be the optimal decision for product j given capacity x_{it}. Then,

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

with terminal condition $\vartheta_{i,\tau+1}^\lambda(\cdot) = 0$. Thus, the value function $\vartheta_{i1}^\lambda(c_i)$ can be written as the expected sum of λ-weighted accepted products:

$$\vartheta_{i1}^\lambda(c_i) = \mathbb{E}\left[\sum_{t'=1}^{\tau} \sum_{j\in\mathcal{J}} p_{jt'} \lambda_{ijt'} y_{ijt'}^{*\lambda}(X_{it'}) \mid X_{i1} = c_i \right]$$

where $X_{it'}$ is the random capacity at time t'.

Capacity State Probability $\mu_{it}(x)$: To see the effect of λ_{ijt} on $\vartheta_{i1}^\lambda(c_i)$, define $\mu_{it}(x)$ as the probability that resource i has capacity x at time t, starting from c_i at $t = 1$ and following the optimal policy $y^{*\lambda}$. This probability mass function is computed recursively:

- **Base case ($t = 1$):** $\mu_{i1}(x) = \mathbb{I}\{x = c_i\}$.
- **Recursive step (for $s = 1, \ldots, \tau - 1$):**

$$\mu_{i,s+1}(x') = \sum_{x=0}^{c_i} \mu_{is}(x) \sum_{j\in\mathcal{J}} p_{js} \mathbb{I}\left\{ x' = x - a_{ij} y_{ijs}^{*\lambda}(x) \right\}$$

The subgradient of $\vartheta_{i1}^\lambda(c_i)$ with respect to λ_{ijt}: As we can see, the change in $\vartheta_{i1}^\lambda(c_i)$ due to λ_{ijt} is the sum of local effects at time t, weighted by the probability $\mu_{it}(x_{it})$ of being in state x_{it}:

$$\frac{\partial \vartheta_{i1}^\lambda(c_i)}{\partial \lambda_{ijt}} = \sum_{x_{it}=0}^{c_i} \mu_{it}(x_{it}) \left(p_{jt} y_{ijt}^{*\lambda}(x_{it}) \right) = p_{jt} \sum_{x_{it}=0}^{c_i} \mu_{it}(x_{it}) y_{ijt}^{*\lambda}(x_{it})$$

Intuitively, the sum $\sum_{x_{it}=0}^{c_i} \mu_{it}(x_{it}) y_{ijt}^{*\lambda}(x_{it})$ is the expected optimal decision for product j at time t, given initial capacity c_i.

The subgradient of $V_1^\lambda(c_1)$ with respect to λ_{ijt}: Using (1), we have:

$$\frac{\partial V_1^\lambda(c_1)}{\partial \lambda_{ijt}} = \frac{\partial \vartheta_{i1}^\lambda(c_i)}{\partial \lambda_{ijt}} - p_{jt} \mathbf{1}\left\{ f_j - \sum_{k\in\mathcal{L}} \lambda_{kjt} \geq 0 \right\}$$

5 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.

5.1 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 $\lambda_{ijt}$
Ensure: Value functions $\vartheta^{\lambda}_{it}(x_{it})$ and optimal decisions $y^*_{ijt}(x_{it})$
 1: Initialize $\vartheta^{\lambda}_{i,\tau+1}(x_{i,\tau+1}) \leftarrow 0$ for all $x_{i,\tau+1} \in \mathcal{C}$        ▷ Terminal condition
 2: for $t = \tau$ down to 1 do        ▷ Backward recursion
 3:     for $x_{it} = 0$ to $C$ do        ▷ For each capacity level
 4:         $\vartheta^{\lambda}_{it}(x_{it}) \leftarrow 0$
 5:         $y^*_{ijt}(x_{it}) \leftarrow 0$ for all $j \in \mathcal{J}$        ▷ Initialize decision variables
 6:         for $j \in \mathcal{J}$ do        ▷ For each product
 7:             if $a_{ij} \le x_{it}$ then        ▷ Check if capacity is sufficient
 8:                 $v_0 \leftarrow \vartheta^{\lambda}_{i,t+1}(x_{it})$        ▷ Value if reject
 9:                 $v_1 \leftarrow \lambda_{ijt} + \vartheta^{\lambda}_{i,t+1}(x_{it} - a_{ij})$        ▷ Value if accept
10:                 if $v_1 > v_0$ then
11:                     $y^*_{ijt}(x_{it}) \leftarrow 1$        ▷ Accept product $j$ at time $t$ with capacity $x_{it}$
12:                 end if
13:             end if
14:         end for
15:         $\vartheta^{\lambda}_{it}(x_{it}) \leftarrow \sum_{j \in \mathcal{J}} p_{jt} \Big[ \lambda_{ijt}\, y^*_{ijt}(x_{it}) + \vartheta^{\lambda}_{i,t+1}\big(x_{it} - a_{ij} y^*_{ijt}(x_{it})\big) \Big]$
16:     end for
17: end for
18: return $\{\vartheta^{\lambda}_{it}(x_{it}) : x_{it} \in \mathcal{C}, t \in \mathcal{T}\}$ and $\{y^*_{ijt}(x_{it}) : j \in \mathcal{J}, x_{it} \in \mathcal{C}, t \in \mathcal{T}\}$
```

5.2 Subroutine: Computing State Probabilities

This subroutine is executed for each resource $i \in \mathcal{L}$. It computes the state occupancy probabilities $\mu_{is}(x)$ for resource i at each time $s \in \mathcal{T}$ and capacity level $x \in \mathcal{C}$. These probabilities indicate the likelihood of resource i having x units of capacity at the beginning of period s, given an initial capacity $c_{i,1}$ at $s = 1$ and following the optimal single-resource policies $y^{*\lambda}_{ijs}(x)$ derived from Algorithm 1.

Algorithm 2 Compute State Probabilities $\mu_{is}(x)$ for Resource i

```
Require: Resource $i$, initial capacity $c_{i,1}$, maximum capacity $C$, time periods $\mathcal{T}$, product set $\mathcal{J}$, arrival probabilities $p_{js}$, consumption $a_{ij}$, optimal policies $y^{*\lambda}_{ijs}(x)$
Ensure: State probabilities $\mu_{is}(x)$ for $s \in \mathcal{T}, x \in \{0, \dots, C\}$
 1: Initialize array $\mu_{i\cdot(\cdot)} : (\mathcal{T} \times \mathcal{C}) \to \mathbb{R}$ with all entries 0.0.
 2: $\mu_{i1}(c_{i,1}) \leftarrow 1.0$                          ▷ At $s = 1$, capacity is $c_{i,1}$ with prob. 1
 3: for $s = 1$ to $\tau - 1$ do                                          ▷ Forward in time
 4:     for $x_{curr} = 0$ to $C$ do                              ▷ For each capacity at $s$
 5:         if $\mu_{is}(x_{curr}) > 10^{-9}$ then                       ▷ If $x_{curr}$ is reachable
 6:             for $j \in \mathcal{J}$ do                                ▷ For each product $j$
 7:                 $y^*_{curr} \leftarrow y^{*\lambda}_{i,j,s}(x_{curr})$           ▷ Optimal decision at $(i, s, x_{curr})$
 8:                 $x_{next} \leftarrow x_{curr} - a_{i,j} \cdot y^*_{curr}$
 9:                 if $x_{next} < 0$ then          ▷ Negative capacity should not actually occur!
10:                     $x_{next} \leftarrow 0$
11:                 end if
12:                 $\mu_{i,s+1}(x_{next}) \leftarrow \mu_{i,s+1}(x_{next}) + \mu_{is}(x_{curr}) \cdot p_{j,s}$
13:             end for
14:         end if
15:     end for
16: end for
17: return $\mu$ array (containing $\mu_{is}(x)$)
```

5.3 Subroutine: Computing ϑ-Subgradient

This subroutine uses the state probabilities and the optimal policies to calculate the subgradient of the single-resource total expected λ-weighted value $\vartheta^{\lambda}_{i1}(c_{i,1})$ with respect to each relevant Lagrange multiplier λ_{ijs}.

Algorithm 3 Compute ϑ-Subgradient G_{ijs} for Resource i

```
Require: State probabilities $\mu_{is}(x)$, optimal policies $y^{*\lambda}_{ijs}(x)$, arrival probabilities $p_{js}$, product set $\mathcal{J}$, time periods $\mathcal{T}$, maximum capacity $C$
Ensure: Subgradients $G_{ijs} = \frac{\partial \vartheta^{\lambda}_{i1}(c_{i,1})}{\partial \lambda_{ijs}}$ for $j \in \mathcal{J}, s \in \mathcal{T}$
 1: Initialize array $G_{i\cdot\cdot} : (\mathcal{J} \times \mathcal{T}) \to \mathbb{R}$ with all entries set to 0.0.
 2: for $s = 1$ to $\tau$ do                                        ▷ For each time period $s$
 3:     for $j \in \mathcal{J}$ do                                        ▷ For each product $j$
 4:         expected_y_star$_{ijs}$ ← 0
 5:         for $x = 0$ to $C$ do                     ▷ For each capacity level $x$ at time $s$
 6:             if $\mu_{is}(x) > 10^{-9}$ then                       ▷ If state $(i, s, x)$ is reachable
 7:                 expected_y_star$_{ijs}$ ← expected_y_star$_{ijs}$ + $\mu_{is}(x) \cdot y^{*\lambda}_{i,j,s}(x)$
 8:             end if
 9:         end for
10:         $G_{i,j,s} \leftarrow p_{j,s} \cdot$ expected_y_star$_{ijs}$
11:     end for
12: end for
13: return $G$ array (containing $G_{ijs}$)
```

5.4 Final Subgradient Optimization

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

Algorithm 4 Projected Subgradient Descent for Lagrangian Multiplier Optimization

```
Require: Initial multipliers $\lambda^0$, step size $\alpha_0$, tolerance $\epsilon$, maximum iterations $K$
Ensure: Optimized multipliers $\lambda^*$
 1: $k \leftarrow 0$
 2: $V_{\text{prev}} \leftarrow \infty$
 3: while $k < K$ do
 4:     Compute $V_1^{\lambda^k}(c_1)$
 5:     if $|V_1^{\lambda^k}(c_1) - V_{\text{prev}}| < \epsilon$ then
 6:         break
 7:     end if
 8:     $V_{\text{prev}} \leftarrow V_1^{\lambda^k}(c_1)$
 9:     Compute subgradient $g^k$ of $V_1^{\lambda^k}(c_1)$ with respect to $\lambda$.
10:     $\alpha_k \leftarrow \frac{\alpha_0}{\sqrt{k+1}}$
11:     $\lambda^{k+1} \leftarrow \max\{0, \lambda^k - \alpha_k g^k\}$          ▷ Project onto feasible set
12:     $k \leftarrow k + 1$
13: end while
14: return $\lambda^k$
```

5.5 Algorithm: Computing Total Expected Revenue using Bid Prices

Once the optimal Lagrangian multipliers λ^* are obtained (e.g., using Algorithm 4), the control policy based on bid prices can be defined. To evaluate the performance of this policy, we can estimate the total expected revenue by simulating customer arrivals and decisions over the time horizon. The following algorithm describes this Monte Carlo simulation process.

Note: Due to the curse of dimensionality, the exact dynamic programming approach is computationally feasible only for small networks. For larger instances, Monte Carlo simulation can approximate the expected revenue by sampling customer arrivals and applying the bid price policy iteratively.

Algorithm 5 Estimate Total Expected Revenue using Bid Price Policy (Monte Carlo Simulation)

```
Input:
  Optimized Lagrangian multipliers $\lambda^*$ (from Algorithm 4)
  Initial capacities $c = (c_1, \dots, c_{|\mathcal{L}|})$ for resources $i \in \mathcal{L}$
  Set of products $\mathcal{J}$ (assumed to include a dummy product $\psi$ with $f_\psi = 0, a_{i\psi} = 0$)
  Product revenues $f_j$ for $j \in \mathcal{J}$
  Resource consumption $a_{ij}$ for $i \in \mathcal{L}, j \in \mathcal{J}$
  Arrival probabilities $p_{jt}$ for $j \in \mathcal{J}, t \in \mathcal{T}$ (with $\sum_{j \in \mathcal{J}} p_{jt} = 1$ for each $t$)
  Time horizon $\mathcal{T} = \{1, \dots, \tau\}$
  Maximum capacity $C$ for any resource (needed for $\vartheta$ table dimensions)
  Number of simulation runs $N_{sim}$
Output:
  Estimated total expected revenue $E[R]$

  Phase 1: Pre-computation
  Compute value functions $\vartheta^{\lambda^*}_{i,t'}(x)$ for all $i \in \mathcal{L}$, $t' \in \{1, \dots, \tau+1\}$, and $x \in \{0, \dots, C\}$ using Algorithm 1 with the input $\lambda^*$.
      ▷ Recall that $\vartheta^{\lambda^*}_{i,\tau+1}(x) = 0$ for all $x$ is the terminal condition for Algorithm 1.

  Phase 2: Monte Carlo Simulation
  Initialize $TotalSimulatedRevenue \leftarrow 0.0$
  for $s = 1$ to $N_{sim}$ do                                   ▷ Iterate over simulation runs
      $CurrentRunRevenue \leftarrow 0.0$
      $current\_capacities \leftarrow$ copy of $c$   ▷ Initialize resource capacities for this run, array indexed by $i$
      for $t = 1$ to $\tau$ do                                   ▷ Iterate over time periods
          Sample a product $j_{req} \in \mathcal{J}$ based on the probability distribution $\{p_{jt}\}_{j \in \mathcal{J}}$ for period $t$.
          if $f_{j_{req}} > 0$ then        ▷ Process actual products; dummy product $\psi$ has $f_\psi = 0$
              $OpportunityCost \leftarrow 0.0$
              $SufficientCapacity \leftarrow$ true
                  ▷ Check capacity and calculate total opportunity cost for product $j_{req}$
              for $i \in \mathcal{L}$ such that $a_{ij_{req}} > 0$ do        ▷ For each resource $i$ consumed by $j_{req}$
                  if $current\_capacities[i] < a_{ij_{req}}$ then
                      $SufficientCapacity \leftarrow$ false
                      break     ▷ Out of loop checking resources for $j_{req}$; insufficient capacity for resource $i$
                  end if
                  for $r = 1$ to $a_{ij_{req}}$ do     ▷ Sum bid prices for each unit of resource $i$ consumed
                      bid_price_unit $\leftarrow \vartheta^{\lambda^*}_{i,t+1}(current\_capacities[i] - r + 1) - \vartheta^{\lambda^*}_{i,t+1}(current\_capacities[i] - r)$
                      $OpportunityCost \leftarrow OpportunityCost +$ bid_price_unit
                  end for
              end for
              if $SufficientCapacity$ and $f_{j_{req}} \geq OpportunityCost$ then    ▷ Acceptance decision
                  $CurrentRunRevenue \leftarrow CurrentRunRevenue + f_{j_{req}}$
                  for $i \in \mathcal{L}$ such that $a_{ij_{req}} > 0$ do              ▷ Update capacities
                      $current\_capacities[i] \leftarrow current\_capacities[i] - a_{ij_{req}}$
                  end for
              end if
          end if     ▷ Capacities in $current\_capacities$ are now updated for the start of the next period $t+1$
      end for                                   ▷ End of time horizon for one simulation run
      $TotalSimulatedRevenue \leftarrow TotalSimulatedRevenue + CurrentRunRevenue$
```

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.