Optimization Letters (2023) 17:437–451
https://doi.org/10.1007/s11590-022-01945-y

ORIGINAL PAPER

Efficient compact linear programs for network revenue management

Simon Laumer[1]

Received: 17 June 2022 / Accepted: 7 October 2022 / Published online: 22 October 2022
© The Author(s) 2022

Abstract
We are concerned with computing bid prices in network revenue management using approximate linear programming. It is well-known that affine value function approximations yield bid prices which are not sensitive to remaining capacity. The analytic reduction to compact linear programs allows the efficient computation of such bid prices. On the other hand, capacity-dependent bid prices can be obtained using separable piecewise linear value function approximations. Even though compact linear programs have been derived for this case also, they are still computationally much more expensive compared to using affine functions. We propose compact linear programs requiring substantially smaller computing times while, simultaneously, significantly improving the performance of capacity-independent bid prices. This simplification is achieved by taking into account remaining capacity only if it becomes scarce. Although our proposed linear programs are relaxations of the unreduced approximate linear programs, we conjecture equivalence and provide according numerical support. We measure the quality of an approximation by the difference between the expected performance of an induced policy and the corresponding theoretical upper bound. Using this paradigm in numerical experiments, we demonstrate the competitiveness of our proposed linear programs.

Keywords Network revenue management · Approximate dynamic programming · Approximate linear programming · Reductions

1 Introduction and literature review

Network revenue management is concerned with the sale of multiple products using multiple perishable resources of finite capacity over a discrete time horizon. The standard reference for network revenue management is Talluri and van

✉ Simon Laumer
simon.laumer@business.uzh.ch

[1] Department of Business Administration, University of Zurich, Plattenstrasse 14, 8032 Zürich, Switzerland

Ryzin [11]. Traditional models assume that for each product, demand does not depend on the availability of other products [1]. This restrictive assumption has been relaxed by introducing customer choice models [10]. One special case of a customer choice model is discrete pricing [3, 4]. Since discrete pricing problems can be reformulated as independent demand problems [16], assuming independent demand is not as restrictive as previously thought.

Determining an optimal control policy in network revenue management requires computing the value function using dynamic programming. Since capacity control involves multiple resources, the curse of dimensionality prohibits the exact computation of the value function. One stream of literature utilizes approximate linear programming to find approximate solutions. The dynamic programming recursion is reformulated as an exponentially large linear program. Then, a value function approximation based on a small number of basis functions is inserted into the linear program [9]. This way, the number of variables is reduced. However, this procedure does not decrease the number of constraints. To overcome this problem, algorithmic techniques such as column generation, constraint sampling, and constraint-violation learning are typically applied [1, 2, 7]. These approaches do not solve the linear program exactly but provide an approximation. It is thus preferable to find reformulations that can be solved directly using a commercial solver. This motivates the derivation of compact linear programs [12, 14]. In this paper, we call a linear program compact if the number of variables and constraints is polynomial in the number of resources, products, time steps and units of initial capacity, which means it is computationally tractable.

An important aspect of approximate linear programming is the choice of basis functions. Choosing basis functions which are separable across resources is appealing, and thus affine and separable piecewise linear functional approximations have received much attention in the literature [1, 4–6]. We refer to these two approximation types as AF and SPL, respectively. Compact reformulations have been derived for both types [12, 14]. While SPL bid prices depend on remaining capacity, the opposite is true for AF bid prices. The components of the SPL approximation are piecewise linear on equidistant grids where the distance between nodes is exactly 1. Consequently, the number of nodes equals the initial capacity size. In contrast, Meissner and Strauss [8] use piecewise linear value function approximations where the number and position of nodes may be chosen arbitrarily. We call this approximation type separable "genuinely" piecewise linear (SGPL) in order to distinguish it from SPL. To the best of our knowledge, there is no study comparing the computational efficiency of AF and SPL with SGPL using compact linear programs. In particular, Meissner and Strauss [8] apply column generation and do not provide compact linear programs.

We make the following contributions:

1. For the independent demand model for network revenue management, we heuristically propose novel compact linear programs which are smaller than the SPL reduction yet improve the quality of AF bid prices by taking into account remaining capacity whenever it becomes scarce.

2. We benchmark our proposed compact linear programs against the reductions for AF and SPL using network instances from the literature [13, 14]. We find that for many instances, AF's optimality gap can be divided in half using significantly less than half of SPL's computing time.

Outline In Sect. 2, we describe the underlying network revenue management model and recapitulate the compact linear programs for AF and SPL. We then propose compact linear programs associated with SGPL basis functions in Sect. 3. Although we do not prove equivalence of our proposed linear programs with their unreduced counterparts, we provide numerical support for a corresponding conjecture in Appendix A. In Sect. 4, we investigate the computational efficiency of SGPL by benchmarking it against AF and SPL.

2 Approximate linear programming in network revenue management

Model description Our model follows Adelman [1]. During a selling horizon of finitely many time steps $t \in \{1, \dots, T\}$, a company sells multiple products $j \in \{1, \dots, J\}$ with fares f_j. We assume that at most one customer arrives per time step t. The probability that at time t product j is requested is denoted $p_{t,j}$. We assume product $j = 1$ to be a dummy product representing the event that no customer arrives, which implies $\sum_j p_{t,j} = 1, \forall t$. At the beginning of each time step t, the company must decide which requests will be accepted and which will be rejected. This decision is represented by the decision vector $(u_j) = \mathbf{u} \in \{0, 1\}^J$.

There are multiple resources $i \in \{1, \dots, I\}$ each of which may be used by several products. The consumption matrix $\mathbf{A} = (a_{ij}) \in \{0, 1\}^{I \times J}$ has corresponding entries: $a_{ij} = 1$ if product j uses one unit of resource i, and $a_{ij} = 0$ otherwise. A column $\mathbf{A}^j$ thus corresponds to the set of resources used by product j. The vector $\mathbf{c} = (c_1, \dots, c_I)^T \in \mathbb{N}^I$ denotes the initial capacity at the beginning of the selling horizon. During the selling process, the remaining capacity is denoted by $\mathbf{r} = (r_1, \dots, r_I)^T$. At time $T + 1$, all of the remaining capacity becomes worthless.

Let $\mathcal{U}_\mathbf{r} = \{\mathbf{u} \in \{0, 1\}^J \mid \forall i, j : u_j a_{ij} \leq r_i\}$ be the set of feasible decision vectors given the remaining capacity $\mathbf{r}$. Furthermore, let

$$\mathcal{R}_t = \begin{cases} \prod_i \{c_i\}, & t = 1 \\ \prod_i \{0, \dots, c_i\}, & t \geq 2 \end{cases}$$

denote the state space at time t. The expected future revenue from time t on, given remaining capacity $\mathbf{r}$ and using an optimal policy, is denoted by the (so-called) value function $v_t(\mathbf{r})$. This function is recursively defined by the Bellman equation

$$
\begin{aligned}
v_t(\mathbf{r}) &= \max_{\mathbf{u}\in\mathcal{U}_\mathbf{r}} \sum_j u_j p_{t,j} \left[f_j - \left[v_{t+1}(\mathbf{r}) - v_{t+1}(\mathbf{r}-\mathbf{A}^j)\right]\right] \\
&\qquad + v_{t+1}(\mathbf{r}), \quad \forall t, \mathbf{r} \in \mathcal{R}_t \\
v_{T+1}(\mathbf{r}) &= 0, \quad \forall \mathbf{r} \in \mathcal{R}_{T+1}.
\end{aligned} \tag{1}
$$

The key element in this dynamic program is the term $f_j - [v_{t+1}(\mathbf{r}) - v_{t+1}(\mathbf{r} - \mathbf{A}^j)]$, i.e., the difference between the revenue f_j resulting from the potential sale of product j at time t, and the marginal value $v_{t+1}(\mathbf{r}) - v_{t+1}(\mathbf{r} - \mathbf{A}^j)$ of product j at time $t+1$. An optimal policy accepts a request for product j if and only if its marginal value does not exceed the fare f_j.

The recursion (1) suffers from the curse of dimensionality. In particular, there are exponentially many values $v_t(\mathbf{r})$ that have to be computed. We therefore turn to the construction of approximate solutions.

Approximate linear programming It is well-known that $v_1(\mathbf{c})$ defined by the Bellman equation (1) is the optimal value of the following linear program:

$$
\begin{aligned}
(D) \quad & \min_v v_1(\mathbf{c}) \\
\text{s.t.} \quad & v_t(\mathbf{r}) \geq \sum_j u_j p_{t,j} \left[f_j - \left[v_{t+1}(\mathbf{r}) - v_{t+1}(\mathbf{r}-\mathbf{A}^j)\right]\right] \\
& \qquad + v_{t+1}(\mathbf{r}), \quad \forall t, \mathbf{r} \in \mathcal{R}_t, \mathbf{u} \in \mathcal{U}_\mathbf{r} \\
& v_{T+1}(\mathbf{r}) = 0, \quad \forall \mathbf{r} \in \mathcal{R}_{T+1}.
\end{aligned}
$$

The size of (D) is growing exponentially in the number of both resources and products. We choose a small number of basis functions $\phi_b(\mathbf{r}), b \in \mathcal{B}$, and insert the value function approximation $v_t(\mathbf{r}) \approx \sum_{b\in\mathcal{B}} V_{t,b}\phi_b(\mathbf{r})$ into (D) to obtain the approximate linear program (D_ϕ). This reduces the number of variables to $(T+1)|\mathcal{B}|$. The corresponding dual is given by:

$$
\begin{aligned}
(P_\phi) \quad & \max_{X\geq 0} \sum_{t,\mathbf{r}\in\mathcal{R}_t,\mathbf{u}\in\mathcal{U}_\mathbf{r}} \sum_j X_{t,\mathbf{r},\mathbf{u}} u_j p_{t,j} f_j \\
\text{s.t.} \quad & \sum_{\mathbf{r}\in\mathcal{R}_t,\mathbf{u}\in\mathcal{U}_\mathbf{r}} X_{t,\mathbf{r},\mathbf{u}}\phi_b(\mathbf{r}) \\
& = \begin{cases} \phi_b(\mathbf{c}), & t = 1 \\ \displaystyle\sum_{\mathbf{r}\in\mathcal{R}_{t-1},\mathbf{u}\in\mathcal{U}_\mathbf{r}} X_{t-1,\mathbf{r},\mathbf{u}}\phi_b(\mathbf{r}) \\ \quad \displaystyle - \sum_{\mathbf{r}\in\mathcal{R}_{t-1},\mathbf{u}\in\mathcal{U}_\mathbf{r}} X_{t-1,\mathbf{r},\mathbf{u}} \sum_j u_j p_{t-1,j} \left[\phi_b(\mathbf{r}) - \phi_b(\mathbf{r}-\mathbf{A}^j)\right], & t \geq 2 \end{cases} \\
& \quad \forall t, b \in \mathcal{B}.
\end{aligned} \tag{2}
$$

The optimal value of (P_ϕ) is an upper bound on $v_1(\mathbf{c})$. If the set of basis functions includes a constant function $\phi_b(\cdot) \equiv 1$, we can show by induction that the property $\sum_{\mathbf{r},\mathbf{u}} X_{t,\mathbf{r},\mathbf{u}} = 1, \forall t$, holds for any feasible solution to (P_ϕ). This observation allows us to interpret each value $X_{t,\mathbf{r},\mathbf{u}}$ as the probability that at time t, the remaining capacity is $\mathbf{r}$ and the decision is $\mathbf{u}$.

Compact linear programs from the literature For the AF approximation $v_t(\mathbf{r}) \approx \theta_t + \sum_i V_{t,i} r_i$, Tong and Topaloglu [12] as well as Vossen and Zhang [14] show equivalence between (P_ϕ) and the compact linear program

$$
\begin{aligned}
(\widehat{P}_{AF}) \quad & \max_{\rho,\mu \geq 0} \sum_{t,j} p_{t,j} \quad f_j \mu_{t,j} \\
\text{s.t.} \quad & \rho_{t,i} = \begin{cases} c_i, & \text{if } t = 1 \\ \rho_{t-1,i} - \sum_{j \text{ s.t. } a_{ij}=1} p_{t-1,j} \mu_{t,j}, & \text{if } t > 1, \end{cases} \quad \forall t, i \\
& \mu_{t,j} \leq \rho_{t,i}, \quad \forall t, i, j : a_{ij} = 1 \\
& \mu_{t,j} \leq 1, \quad \forall t, j.
\end{aligned}
$$

Similar to the interpretation of $X_{t,\mathbf{r},\mathbf{u}}$ as state-action probabilities, $\mu_{t,j}$ represents the probability of decision $u_j = 1$ in time step t. The variable $\rho_{t,i}$ is an approximation of the expected value of r_i at time t.

For the SPL approximation $v_t(\mathbf{r}) \approx \theta_t + \sum_i \sum_{k=1}^{c_i} V_{t,i,k} 1_{\{r_i \geq k\}}$, where $1_{\{r_i \geq k\}}$ denotes the indicator function, Vossen and Zhang [14] show weak equivalence between (P_ϕ) and the compact linear program

$$
\begin{aligned}
(\widehat{P}_{SPL}) \quad & \max_{\zeta,\sigma,\mu \geq 0} \sum_{t,j} p_{t,j} f_j \mu_{t,j} \\
\text{s.t.} \quad & \sigma_{t,i,k} = \begin{cases} 1, & t = 1 \\ \sigma_{t-1,i,k} - \sum_{j \text{ s.t. } a_{ij}=1} p_{t-1,j} (\zeta_{t-1,i,j,k} - \zeta_{t-1,i,j,k+1}) & t \geq 2 \end{cases} \quad \forall t, i, k \qquad (3) \\
& \mu_{t,j} = \zeta_{t,i,j,1}, \quad \forall t, i, j : a_{ij} = 1 \qquad (4) \\
& \zeta_{t,i,j,k+1} \leq \zeta_{t,i,j,k}, \quad \forall t, i, j, k : a_{ij} = 1 \qquad (5) \\
& \zeta_{t,i,j,k} \leq \sigma_{t,i,k}, \quad \forall t, i, j, k : a_{ij} = 1. \qquad (6)
\end{aligned}
$$

The interpretation of $\mu_{t,j}$ is the same as above. The variable $\sigma_{t,i,k}$ represents the probability that at time t, resource i has at least k units left. $\zeta_{t,i,j,k}$ represents the joint probability that at time t, resource i has at least k units left and the decision $u_j = 1$ is made. To enforce the probabilistic interpretation of σ, one would expect that $(\widehat{P}_{SPL})$ includes the constraints $\sigma_{t,i,1} \leq 1, \forall t, i$ and $\sigma_{t,i,k+1} \leq \sigma_{t,i,k}, \forall t, i, k$. However, these constraints are redundant [14].

3 Genuinely piecewise linear approximation

We intend to decrease the size of $(\widehat{P}_{SPL})$ by considering genuinely piecewise linear functions where the number of nodes can be chosen arbitrarily. Concerning the position of the nodes, we remember that revenue management is most crucial whenever remaining capacity becomes scarce. Separately for each resource, our proposed value function approximation is thus piecewise linear with nodes $0, 1, 2, \dots, L_i - 1, c_i$, where $L_i \in \mathbb{N}$ satisfies $1 \leq L_i \leq c_i$. Piecewise linear functions on this grid are spanned by the basis functions $1_{\{r_i \geq k\}}, k = 1, \dots, L_i - 1$, together with the additional basis function $\max\{0, r_i - L_i + 1\}$. Therefore, our proposed value function approximation has the following form:

$$v_t(\mathbf{r}) \approx \theta_t + \sum_i \left(\sum_{k=1}^{L_i - 1} V_{t,i,k} 1_{\{r_i \geq k\}} + V_{t,i,L_i} \max\{0, r_i - L_i + 1\} \right). \tag{7}$$

We now develop a compact linear program denoted $(\overline{P}_G)$ associated with the value function approximation (7). This is done heuristically by modifying $(\widehat{P}_{SPL})$.

For a fixed time t, each constraint in (3) corresponds to a basis function $1_{\{r_i \geq k\}}$. Since the value function approximation (7) includes the basis functions $1_{\{r_i \geq k\}}$ for $k \leq L_i - 1$, $(\overline{P}_G)$ inherits the constraints (3) for $k \leq L_i - 1$. Our main task is to construct analogue constraints for the basis functions $\max\{0, r_i - L_i + 1\}, i = 1, \dots, I$. We first observe that the left hand side of (3), $\sigma_{t,i,k}$, corresponds to the unreduced term $\sum_{\mathbf{r} \in \mathcal{R}_t, \mathbf{u} \in \mathcal{U}_\mathbf{r}} X_{t,\mathbf{r},\mathbf{u}} 1_{\{r_i \geq k\}}$ on the left hand side of (2). This term is the expected value of $1_{\{r_i \geq k\}}$ given the probability distribution X_t. Adapting this probabilistic view for the basis function $\max\{0, r_i - L_i + 1\}$, we look for the expected value of $\max\{0, r_i - L_i + 1\}$ given the probability distribution X_t. In terms of the probabilities $\sigma_{t,i,k}$, this translates to $\sum_{k=L_i}^{c_i} (\sigma_{t,i,k} - \sigma_{t,i,k+1})(k - L_i + 1) = \sum_{k=L_i}^{c_i} \sigma_{t,i,k}$. The fact that we end up with the sum of $\sigma_{t,i,k}$ over $k = L_i, \dots, c_i$ suggests that the constraints we intend to construct result from summing (3) over $k = L_i, \dots, c_i$:

$$\sum_{k=L_i}^{c_i} \sigma_{t,i,k} = \sum_{k=L_i}^{c_i} \sigma_{t-1,i,k} - \sum_{\substack{j \\ \text{s.t. } a_{ij} = 1}} p_{t-1,j} \zeta_{t-1,i,j,L_i}, \quad \forall t, i. \tag{8}$$

Here, ζ_{t-1,i,j,L_i} on the right hand side is the result of a telescoping sum and the fact that $\zeta_{t,i,j,c_i+1} = 0$. Summarized, we obtain $(\overline{P}_G)$ as a relaxation of $(\widehat{P}_{SPL})$ by replacing the constraints (3) for $k = L_i, \dots, c_i$ with their sum. This constraint aggregation decreases the number of constraints and is thus a simplification of $(\widehat{P}_{SPL})$. To decrease the number of variables, we use the probabilistic interpretation of σ to argue as follows: For a given number σ_{t,i,L_i}, the term $\sum_{k=L_i}^{c_i} \sigma_{t,i,k}$ can take any value between σ_{t,i,L_i} and $\sigma_{t,i,L_i}(c_i - L_i + 1)$. The same is true for the term $\sigma_{t,i,L_i} + \sigma_{t,i,c_i}(c_i - L_i)$. The variables $\sigma_{t,i,L_i+1}, \dots, \sigma_{t,i,c_i-1}$ are thus superfluous, and the above constraints (8) become

$$\sigma_{t,i,L_i} + \sigma_{t,i,c_i}(c_i - L_i) = \sigma_{t-1,i,L_i} + \sigma_{t-1,i,c_i}(c_i - L_i) - \sum_{\substack{j \\ \text{s.t. } a_{ij}=1}} p_{t-1,j}\zeta_{t-1,i,j,L_i}, \quad \forall t, i.$$

Finally, we add the constraints $\sigma_{t,i,k} \le 1$ and $\sigma_{t,i,k+1} \le \sigma_{t,i,k}$ which were redundant for $(\widehat{P}_{SPL})$, and propose the following compact linear program:

$$(\overline{P}_G) \quad \max_{\zeta,\mu,\sigma \ge 0} \sum_{t,j} p_{t,j} f_j \mu_{t,j}$$

$$\text{s.t.} \quad \sigma_{t,i,k} = \begin{cases} 1, & t = 1 \\ \sigma_{t-1,i,k} - \sum_{\substack{j \\ \text{s.t. } a_{ij}=1}} p_{t-1,j}(\zeta_{t-1,i,j,k} - \zeta_{t-1,i,j,k+1}), & t \ge 2 \end{cases} \quad \forall t, i, k \le L_i - 1$$

$$\sigma_{t,i,L_i} + \sigma_{t,i,c_i}(c_i - L_i) = \begin{cases} c_i - L_i + 1, & t = 1 \\ \sigma_{t-1,i,L_i} + \sigma_{t-1,i,c_i}(c_i - L_i) - \sum_{\substack{j \\ \text{s.t. } a_{ij}=1}} p_{t-1,j}\zeta_{t-1,i,j,L_i}, & t \ge 2 \end{cases} \quad \forall t, i$$

$$\begin{aligned} \sigma_{t,i,1} &\le 1, && \forall t, i \\ \mu_{t,j} &= \zeta_{t,i,j,1}, && \forall t, i, j : a_{ij} = 1 \\ \zeta_{t,i,j,k+1} &\le \zeta_{t,i,j,k}, && \forall t, i, j, k \le L_i - 1 : a_{ij} = 1 \\ \zeta_{t,i,j,k} &\le \sigma_{t,i,k}, && \forall t, i, j, k \in \{1, \ldots, L_i, c_i\} : a_{ij} = 1 \\ \sigma_{t,i,k+1} &\le \sigma_{t,i,k}, && \forall t, i, k \le L_i - 1 \\ \sigma_{t,i,c_i} &\le \sigma_{t,i,L_i}, && \forall t, i. \end{aligned}$$

Let (P_G) denote the linear program (P_ϕ) using the SGPL approximation (7), and let $\overline{Z}_G$ and Z_G be the optimal values of $(\overline{P}_G)$ and (P_G). It follows from the above discussion that $\overline{Z}_G$ decreases as the number of nodes, L_i, increases. Furthermore, standard arguments from variable aggregation show that the inequality $\overline{Z}_G \ge Z_G$ holds. We conjecture that $(\overline{P}_G)$ is indeed a reduction of (P_G) meaning that $\overline{Z}_G = Z_G$. In any case, $\overline{Z}_G$ provides an upper bound on the optimal expected revenue, i.e., $\overline{Z}_G \ge v_1(\mathbf{c})$.

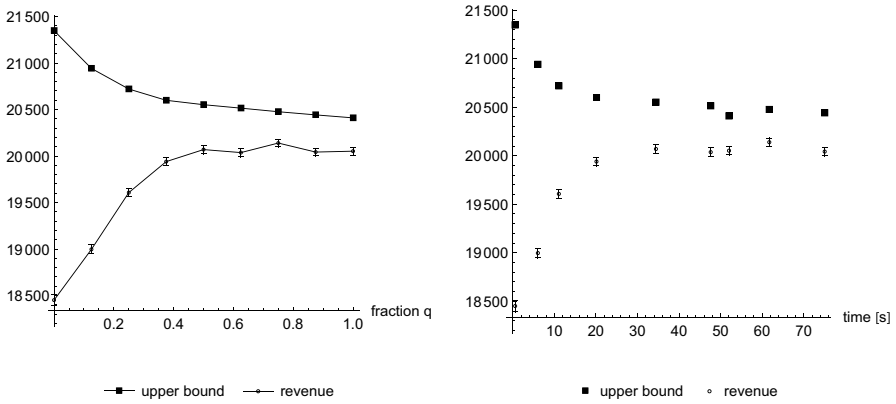

Fig. 1 Upper bounds and average revenues for the network instance $(T, N, \alpha, \kappa) = (200, 4, 1.0, 4)$

To support our conjecture $\overline{Z}_G = Z_G$, we compare these two values on small random network instances, see Appendix A for details. The corresponding AMPL code is available on GitHub so that our results can be reproduced.[1]

4 Numerical experiments

We experimentally benchmark SGPL against AF and SPL. We expect that SGPL outperforms AF and needs less computing time than SPL. It is not clear, however, how fast the quality improves as the number of nodes, L_i, increases. We provide some guidance on how to choose the number of nodes, L_i, during deployment where solving SPL is computationally too expensive.

All numerical experiments were carried out on a virtual machine with 256 GB RAM and 32 cores of 2.59 GHz processors. The linear programs are solved with CPLEX 20.1.0.0, using the interior-point solver "barrier" with standard tolerance 10^{-8}.

We use data files[2] from the literature containing 48 network instances [13, 14]. The setup is a hub-and-spoke airline network with one single hub and N non-hub locations. Each non-hub location is connected with the hub via two legs, one for each direction. There are $N(N+1)$ itineraries corresponding to all origin–destination pairs. For each itinerary, there are two fare classes, where the high fare is κ times higher than the low fare. Therefore, including the dummy product, there is a total of $2N(N+1)+1$ products. Finally, $\alpha := \frac{\sum_{t,i,j} a_{ij} p_{t,j}}{\sum_i c_i}$ denotes the total load. Each network instance is identified with the tupel (T, N, α, κ).

[1] https://github.com/slaume/SGPL-Support-Equivalence-Conjecture.
[2] https://people.orie.cornell.edu/huseyin/research/rm_datasets/rm_datasets.html.

Table 1 Dividing in half AF's optimality gap: Upper bounds (UB) and average revenues for AF, SGPL and SPL

Network	AF		SGPL		SPL		Max.
Instance	UB	Revenue	UB	Revenue	UB	Revenue	Std. err.
(200, 4, 1.0, 4)	21,348	18,451	20,720	19,607	20,411	20,052	58
(200, 4, 1.0, 8)	34,384	28,480	33,572	31,412	33,229	32,715	124
(200, 4, 1.2, 4)	19,663	14,069	19,080	16,805	18,856	18,426	48
(200, 4, 1.2, 8)	32,696	20,215	31,935	26,781	31,614	31,099	103
(200, 4, 1.6, 4)	17,303	11,006	16,809	13,860	16,507	16,123	43
(200, 4, 1.6, 8)	30,335	15,884	29,691	23,547	29,208	28,713	92
(200, 5, 1.0, 4)	22,016	18,419	21,567	20,310	21,257	21,045	58
(200, 5, 1.0, 8)	35,258	28,045	34,718	32,389	34,323	34,131	118
(200, 5, 1.2, 4)	21,108	16,339	20,510	18,845	20,089	19,739	59
(200, 5, 1.2, 8)	34,329	28,587	33,513	30,820	33,027	32,740	121
(200, 5, 1.6, 4)	18,565	14,245	17,750	16,828	17,625	17,183	50
(200, 5, 1.6, 8)	31,758	25,600	30,689	28,503	30,457	29,938	117
(200, 6, 1.0, 4)	22,116	18,374	21,475	20,028	21,075	20,688	57
(200, 6, 1.0, 8)	35,353	27,674	34,534	32,282	34,058	33,626	118
(200, 6, 1.2, 4)	20,649	16,845	19,971	18,409	19,601	19,140	58
(200, 6, 1.2, 8)	33,869	26,713	32,705	31,323	32,474	32,044	125
(200, 6, 1.6, 4)	18,230	13,525	17,599	15,316	17,227	16,766	51
(200, 6, 1.6, 8)	31,436	19,757	30,626	25,122	30,024	29,525	102
(200, 8, 1.0, 4)	19,870	15,779	19,256	17,538	18,712	18,127	53
(200, 8, 1.0, 8)	31,641	23,449	30,850	27,944	30,198	29,476	110
(200, 8, 1.2, 4)	18,598	13,604	17,925	16,117	17,426	16,827	47
(200, 8, 1.2, 8)	30,353	23,068	29,501	25,869	28,813	28,133	113
(200, 8, 1.6, 4)	16,378	12,691	15,519	14,201	15,241	14,720	48
(200, 8, 1.6, 8)	28,118	19,711	27,306	23,309	26,489	25,828	104
(600, 4, 1.0, 4)	32,213	28,293	31,347	29,988	30,969	30,670	75
(600, 4, 1.0, 8)	51,876	44,010	50,765	48,391	50,371	50,116	165
(600, 4, 1.2, 4)	29,618	21,462	28,804	25,846	28,580	28,224	69
(600, 4, 1.2, 8)	49,279	31,056	48,221	41,521	47,897	47,607	156
(600, 4, 1.6, 4)	26,082	16,502	25,374	20,871	25,050	24,721	63
(600, 4, 1.6, 8)	45,742	23,769	44,825	37,697	44,310	44,011	146
(600, 5, 1.0, 4)	33,153	28,662	32,538	30,864	32,203	32,031	89
(600, 5, 1.0, 8)	53,134	44,111	52,669	48,446	51,983	51,702	190
(600, 5, 1.2, 4)	31,773	24,135	31,343	28,220	30,522	29,986	83
(600, 5, 1.2, 8)	51,717	35,576	51,137	46,958	50,129	49,625	179
(600, 5, 1.6, 4)	28,022	21,702	26,952	24,792	26,816	26,337	74
(600, 5, 1.6, 8)	47,939	35,324	46,813	40,834	46,292	45,915	172
(600, 6, 1.0, 4)	26,722	22,707	25,946	24,494	25,497	25,174	72
(600, 6, 1.0, 8)	42,703	34,739	41,714	39,300	41,199	40,879	154
(600, 6, 1.2, 4)	24,878	19,087	24,036	21,971	23,649	23,216	68
(600, 6, 1.2, 8)	40,834	31,505	39,721	35,753	39,209	38,774	151

Table 1 (continued)

Network	AF		SGPL		SPL		Max.
Instance	UB	Revenue	UB	Revenue	UB	Revenue	Std. err.
(600, 6, 1.6, 4)	21,893	15,728	21,143	18,369	20,731	20,286	61
(600, 6, 1.6, 8)	37,842	28,139	36,584	32,906	36,211	35,706	146
(600, 8, 1.0, 4)	23,998	19,777	23,306	21,534	22,704	22,110	67
(600, 8, 1.0, 8)	38,217	30,005	37,326	34,229	36,612	35,918	142
(600, 8, 1.2, 4)	22,382	17,732	21,584	19,482	21,042	20,395	63
(600, 8, 1.2, 8)	36,580	28,095	35,565	31,812	34,820	34,060	139
(600, 8, 1.6, 4)	19,761	14,900	19,021	16,704	18,452	17,814	60
(600, 8, 1.6, 8)	33,942	25,365	32,592	29,305	32,078	31,574	127

It turns out that solving the dual linear programs is computationally more efficient. This observation might be explained by the fact that "barrier" utilizes the matrix product of the constraint matrix and its transpose in each iteration. Since $(\overline{P}_G)$ has approximately twice as many constraints as it has variables, this matrix product is smaller for the dual linear program.

Also, adding the concavity property $V_{t,i,k+1} \leq V_{t,i,k}, \forall t, i, k$, to the constraints of the dual of $(\widehat{P}_{SPL})$ speeds up its computing time. Similar results concerning the increase of efficiency by enforcing concavity of bid prices with respect to time for an affine value function approximation is discussed in [15].

For a fixed network instance (T, N, α, κ), let $\overline{Z}_G^{\mathbf{L}}$ denote the optimal value of $(\overline{P}_G)$ given $\mathbf{L} = (L_1, \ldots, L_I)$. We also call this value the upper bound. Using the value function approximation (7) to compute approximate marginal values, we simulate the corresponding policy 500 times. Let $R_G^{\mathbf{L}}$ denote the resulting average revenue. We measure the quality of an SGPL approximation by the difference $\overline{Z}_G^{\mathbf{L}} - R_G^{\mathbf{L}}$ which we call optimality gap. For a given number $q \in (0, 1)$, we define $L_i^q := \lceil q c_i \rceil, \forall i$. Let t_{comp}^q be the computing time for solving the dual of $(\overline{P}_G)$. Since we always associate $q = 0$ and $q = 1$ with AF and SPL, we report the computing times for solving the dual of $(\widehat{P}_{AF})$ or $(\widehat{P}_{SPL})$ in these cases. We compute $\overline{Z}_G^{\mathbf{L}^q}$ and $R_G^{\mathbf{L}^q}$ for $q = 0, \frac{1}{8}, \ldots, \frac{7}{8}, 1$. Figure 1 shows the results for the network instance $(T, N, \alpha, \kappa) = (200, 4, 1.0, 4)$. On the left hand side, the results are plotted against the fraction q, and on the right hand side, they are plotted against the computing time t_{comp}^q in seconds. We observe large improvements for both the upper bound $\overline{Z}_G^{\mathbf{L}}$ and the average revenue $R_G^{\mathbf{L}}$ even for small fractions q and for computing times which are significantly smaller compared to SPL.

Fixing the value $q = \frac{1}{4}$, SGPL's optimality gap is less than half of AF's optimality gap in 41 out of all 48 network instances. In general, let q_{half} be the smallest $q \in \{\frac{1}{8}, \ldots, \frac{7}{8}, 1\}$ for which the optimality gap is less than half of AF's optimality gap. During deployment, we suggest to either 1) successively solve SGPL for $q = 0, \frac{1}{8}, \ldots, q_{\text{half}}$, or 2) use the fixed value $q = \frac{1}{4}$. In Tables 1 and 2, we report

Table 2 Dividing in half AF's optimality gap: Computing time in seconds for AF, SGPL and SPL

Network		AF	SGPL		SPL
Instance	q_{half}	t^0_{comp}	$t^{q_{\text{half}}}_{\text{comp}}$	$t^0_{\text{comp}} + \cdots + t^{q_{\text{half}}}_{\text{comp}}$	t^1_{comp}
(200, 4, 1.0, 4)	0.25	0.56	11.1	17.7	52.0
(200, 4, 1.0, 8)	0.25	0.50	10.3	15.4	57.7
(200, 4, 1.2, 4)	0.25	0.50	7.6	12.0	36.0
(200, 4, 1.2, 8)	0.25	0.42	7.7	12.1	35.1
(200, 4, 1.6, 4)	0.25	0.45	4.8	7.6	28.8
(200, 4, 1.6, 8)	0.25	0.38	4.9	8.1	28.1
(200, 5, 1.0, 4)	0.25	0.62	19.1	27.8	82.8
(200, 5, 1.0, 8)	0.25	0.52	23.0	30.6	81.6
(200, 5, 1.2, 4)	0.25	0.59	16.6	24.8	55.5
(200, 5, 1.2, 8)	0.25	0.50	14.4	24.1	62.0
(200, 5, 1.6, 4)	0.375	0.64	17.6	30.3	48.3
(200, 5, 1.6, 8)	0.375	0.67	17.5	31.8	52.8
(200, 6, 1.0, 4)	0.25	1.00	21.2	36.8	94.8
(200, 6, 1.0, 8)	0.25	1.20	27.6	43.5	91.5
(200, 6, 1.2, 4)	0.25	1.05	21.3	30.3	86.5
(200, 6, 1.2, 8)	0.375	0.86	40.3	72.8	86.5
(200, 6, 1.6, 4)	0.25	0.97	9.7	15.4	58.7
(200, 6, 1.6, 8)	0.25	0.97	14.6	21.1	64.1
(200, 8, 1.0, 4)	0.25	6.23	33.3	49.3	213.9
(200, 8, 1.0, 8)	0.25	6.84	35.1	54.8	251.3
(200, 8, 1.2, 4)	0.25	7.61	28.8	44.1	350.6
(200, 8, 1.2, 8)	0.25	6.00	29.4	44.1	389.8
(200, 8, 1.6, 4)	0.375	4.30	26.5	50.5	99.5
(200, 8, 1.6, 8)	0.25	4.34	12.7	22.4	108.7
(600, 4, 1.0, 4)	0.25	1.53	97.4	156.2	255.7
(600, 4, 1.0, 8)	0.25	1.16	97.4	157.0	329.3
(600, 4, 1.2, 4)	0.25	1.17	61.0	101.0	239.4
(600, 4, 1.2, 8)	0.25	1.12	56.8	107.3	249.4
(600, 4, 1.6, 4)	0.25	1.11	36.8	56.0	237.3
(600, 4, 1.6, 8)	0.25	1.02	47.0	64.9	252.0
(600, 5, 1.0, 4)	0.25	5.00	128.5	200.0	602.3
(600, 5, 1.0, 8)	0.125	3.31	67.4	70.7	744.1
(600, 5, 1.2, 4)	0.125	2.59	64.0	66.6	383.2
(600, 5, 1.2, 8)	0.125	2.69	59.3	62.0	382.4
(600, 5, 1.6, 4)	0.375	3.09	148.3	260.8	316.1
(600, 5, 1.6, 8)	0.25	4.83	84.9	125.8	365.4
(600, 6, 1.0, 4)	0.25	17.75	138.3	610.0	566.5
(600, 6, 1.0, 8)	0.25	11.84	140.8	483.5	485.4
(600, 6, 1.2, 4)	0.25	9.70	338.0	532.8	343.4
(600, 6, 1.2, 8)	0.25	9.72	184.2	291.0	373.0
(600, 6, 1.6, 4)	0.25	7.20	166.9	221.6	234.3
(600, 6, 1.6, 8)	0.375	6.78	103.4	333.4	207.3

Table 2 (continued)

Network		AF	SGPL		SPL
Instance	q_{half}	t^0_{comp}	$t^{q_{\text{half}}}_{\text{comp}}$	$t^0_{\text{comp}} + \cdots + t^{q_{\text{half}}}_{\text{comp}}$	t^1_{comp}
(600, 8, 1.0, 4)	0.25	43.14	219.8	345.6	1059.4
(600, 8, 1.0, 8)	0.25	42.72	207.5	332.2	1062.5
(600, 8, 1.2, 4)	0.25	69.62	143.3	274.3	675.9
(600, 8, 1.2, 8)	0.25	38.92	146.8	247.8	799.5
(600, 8, 1.6, 4)	0.25	56.84	79.8	223.5	399.0
(600, 8, 1.6, 8)	0.375	92.78	153.2	414.4	456.8

Table 3 Optimality gap and computing times for AF, SPL and SGPL using the fixed value $q = \frac{1}{4}$ for those network instances where $q_{\text{half}} > \frac{1}{4}$

Network	AF		SGPL		SPL		max.
Instance	Gap	t^0_{comp}	Gap	$t^{\frac{1}{4}}_{\text{comp}}$	Gap	t^1_{comp}	Std. err.
(200, 5, 1.6, 4)	4320	0.64	2919	8.0	442	48.3	50
(200, 5, 1.6, 8)	6158	0.67	5157	10.0	519	52.8	117
(200, 6, 1.2, 8)	7156	0.86	4154	16.3	430	86.5	125
(200, 8, 1.6, 4)	3687	4.30	2036	13.7	521	99.5	48
(600, 5, 1.6, 4)	6320	3.09	3917	67.2	479	316.1	74
(600, 6, 1.6, 8)	9703	6.78	6944	199.2	505	207.3	146
(600, 8, 1.6, 8)	8577	92.78	4474	89.5	504	456.8	127

results for strategy 1). Table 1 contains upper bounds and average revenues for AF, SPL and SGPL using q_{half}. Table 2 contains the computing times for AF and SPL, i.e., t^0_{comp} and t^1_{comp}, as well as relevant computing times concerning SGPL: We report both the computing time $t^{q_{\text{half}}}_{\text{comp}}$ as well as the cumulated computing time $t^0_{\text{comp}} + \cdots + t^{q_{\text{half}}}_{\text{comp}}$. To obtain an impression of the qualitative risk associated with strategy 2), Table 3 reports optimality gaps and computing times for AF, SPL and SGPL using the fixed value $q = \frac{1}{4}$ for those instances where $q_{\text{half}} > \frac{1}{4}$.

SGPL's computing time using q_{half} is less than half of SPL's computing time in 46 cases, less than a third in 39 cases, less than a fifth in 17 cases and less than a tenth in 3 cases. The cumulated computing time $t^0_{\text{comp}} + \cdots + t^{q_{\text{half}}}_{\text{comp}}$ is less than half of SPL's computing time in 34 cases, less than a third in 20 cases and less than a fifth in 5 cases. For those cases where $q_{\text{half}} > \frac{1}{4}$, using $q = \frac{1}{4}$ also substantially reduces AF's optimality gap requiring computing times that are significantly smaller than half of SPL's computing time in all but one instance.

5 Conclusion

We add to the literature concerning compact approximate linear programs in network revenue management by filling the gap between the AF and SPL value function approximation. The drawback of AF compared to SPL is mitigated by allowing bid prices to depend on remaining capacity whenever this quantity becomes scarce. At the same time, the computational complexity of SPL is decreased significantly. Our numerical experiments demonstrate that for many instances, AF's optimality gap can be divided in half using only a small fraction of the computing time required to solve SPL.

Further research may be done to extend our work to more general customer choice models. Even though our results can be applied for discrete pricing problems, fields like the retail industry require more sophisticated choice models.

A Numerical support for conjectured equivalence

We provide numerical support for the conjecture $\overline{Z}_G = Z_G$ using small random network instances. The computation of Z_G is made possible by a partial reduction of (P_G), see Appendix B for details.

The network has five nodes, A, B, C, D, E, and four legs, AC, BC, CD, CE. We set $c_i = 7, \forall i$, and $T = 30$. There are eight possible origin–destination pairs, AC, BC, CD, CE, AD, AE, BD, BE, out of which we randomly choose five. For each chosen origin–destination pair, there are two fares which are determined using a uniform distribution over $\{10, \ldots, 30\}$ and $\{40, \ldots, 120\}$, respectively. Demand is stationary and chosen randomly such that $\sum_{t,j} p_{t,j} = T$. The number of nodes $L_i \in \{1, \ldots, c_i\}$ is also chosen randomly for each resource i. We generate twenty such random instances and always observe $\overline{Z}_G = Z_G$.

B Partial reduction

In Appendix A, we have to compute the optimal value of the unreduced linear program (P_G). (P_G) suffers from the curse of dimensionality concerning both the states $\mathbf{r}$ and decisions $\mathbf{u}$. We partially reduce (D_G), the dual of (P_G), concerning the states $\mathbf{r}$. For the sake of simplicity, we abbreviate the SGPL approximation (7) as $\theta_t + \sum_i v^G_{t,i}(r_i)$ where

$$v^G_{t,i}(r_i) := \sum_{k=1}^{L_i - 1} V_{t,i,k} 1_{\{r_i \geq k\}} + V_{t,i,L_i} \max\{0, r_i - L_i + 1\}.$$

Then, the dual of (P_ϕ), which results from inserting (7) into (D), is equal to

$$
\begin{aligned}
(D_G) \quad & \min_{\theta, V} \theta_1 + \sum_i v^G_{1,i}(c_i) \\
\text{s.t.} \quad & \sum_i \left(v^G_{t,i}(r_i) - v^G_{t+1,i}(r_i) + \sum_j u_j p_{t,j} \left(v^G_{t+1,i}(r_i) - v^G_{t+1,i}(r_i - a_{ij}) \right) \right) \\
& \geq \theta_{t+1} - \theta_t + \sum_j u_j p_{t,j} f_j, \quad \forall t, \mathbf{r} \in \mathcal{R}_t, \mathbf{u} \in \mathcal{U}_{\mathbf{r}}
\end{aligned}
$$

with boundary conditions $\theta_{T+1} = V_{T+1,i,k} = 0, \forall i, k$. The unreduced linear program (D_G) is equivalent to the partially reduced linear program

$$
\begin{aligned}
(\widetilde{D}_G) \quad & \min_{\theta, V, \alpha} \theta_1 + \sum_i v^G_{1,i}(c_i) \\
\text{s.t.} \quad & \sum_i \alpha_{t,i,\mathbf{u}} \geq \theta_{t+1} - \theta_t + \sum_j u_j p_{t,j} f_j, \quad \forall t, \mathbf{u} \in \{0,1\}^J
\end{aligned}
\tag{10}
$$

$$
\begin{aligned}
& v^G_{t,i}(r) - v^G_{t+1,i}(r) + \sum_j u_j p_{t,j} \left(v^G_{t+1,i}(r) - v^G_{t+1,i}(r - a_{ij}) \right) \geq \alpha_{t,i,\mathbf{u}}, \\
& \forall t, i, \mathbf{u} \in \{0,1\}^J, r \in \mathcal{R}_{i,\mathbf{u}}
\end{aligned}
\tag{11}
$$

where $\mathcal{R}_{i,\mathbf{u}} := \{r \in \{0, \ldots, c_i\} \mid u_j a_{ij} \leq r, \forall j\}$. Moreover, since $v^G_{t,i}(r)$ is linear for $L_i - 1 \leq r \leq c_i$, the constraints (11) only have to be enforced for $r \in \{0, 1, \ldots, L_i, c_i\} \cap \mathcal{R}_{i,\mathbf{u}}$.

Proof Let θ, V be a feasible solution to (D_G). We obtain a feasible solution θ, V, α to $(\widetilde{D}_G)$ with equal optimal value by defining

$$
\alpha_{t,i,\mathbf{u}} := \min_{r \in \mathcal{R}_{i,\mathbf{u}}} \left(v^G_{t,i}(r) - v^G_{t+1,i}(r) + \sum_j u_j p_{t,j} \left(v^G_{t+1,i}(r) - v^G_{t+1,i}(r - a_{ij}) \right) \right).
$$

Vice versa, given a feasible solution θ, V, α to $(\widetilde{D}_G)$, θ, V is automatically a feasible solution to (D_G) with equal optimal value, which follows from inserting (11) into (10).

Funding Open access funding provided by University of Zurich.

Data availability All datasets analyzed in Sect. 4 are taken from the literature [13] and are available online here: https://people.orie.cornell.edu/huseyin/research/rm_datasets/rm_datasets.html. The AMPL code used in Appendix A is available online on GitHub: https://github.com/slaume/SGPL-Support-Equivalence-Conjecture.

Declarations

Conflict of interest The author declares that he has no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Adelman, D.: Dynamic bid prices in revenue management. Oper. Res. **55**(4), 647–661 (2007)
2. de Farias, D.P., Van Roy, B.: On constraint sampling in the linear programming approach to approximate dynamic programming. Math. Oper. Res. **29**(3), 462–478 (2004)
3. Erdelyi, A., Topaloglu, H.: Using decomposition methods to solve pricing problems in network revenue management. J. Revenue Pricing Manag. **10**(4), 325–343 (2011)
4. Ke, J., Zhang, D., Zheng, H.: An approximate dynamic programming approach to dynamic pricing for network revenue management. Prod. Oper. Manag. **28**(11), 2719–2737 (2019)
5. Kunnumkal, S., Talluri, K.: A note on relaxations of choice network revenue management dynamic program. Oper. Res. **64**(1), 158–166 (2016)
6. Kunnumkal, S., Talluri, K.: On a piecewise-linear approximation for network revenue management. Math. Oper. Res. **41**(1), 72–91 (2016)
7. Lin, Q., Nadarajah, S., Soheili, N.: Revisiting approximate linear programming: Constraint-violation learning with applicatins to inventory control and energy storage. Manag. Sci. **66**(4), 1544–1562 (2019)
8. Meissner, J., Strauss, A.K.: Network revenue management with inventory-sensitive bid prices and customer choice. Eur. J. Oper. Res. **216**(2), 459–468 (2012)
9. Schweitzer, P.J., Seidmann, A.: Generalized polynomial approximations in Markovian decision processes. J. Math. Anal. Appl. **110**(2), 568–582 (1985)
10. Talluri, K., van Ryzin, G.J.: Revenue management under a general discrete choice model of consumer behavior. Manag. Sci. **50**(1), 15–33 (2004)
11. Talluri, K., van Ryzin, G.J.: The Theory and Practice of Revenue Management. Kluwer Academic Publishers, Norwell, MA (2004)
12. Tong, C., Topaloglu, H.: On the approximate linear programming approach for network revenue management problems. INFORMS J. Comput. **26**(1), 121–134 (2014)
13. Topaloglu, H.: Using Lagrangian relaxation to compute capacity-dependent bid prices in network revenue management. Oper. Res. **57**(3), 637–649 (2009)
14. Vossen, T.W.M., Zhang, D.: Reductions of approximate linear programs for network revenue management. Oper. Res. **63**(6), 1352–1371 (2015)
15. Vossen, T.W.M., Zhang, D.: A dynamic disaggregation approach to approximate linear programs for network revenue management. Prod. Oper. Manag. **24**(3), 469–487 (2015)
16. Walczak, D., Mardan, S., Kallesen, R.: Customer choice, fare adjustments and the marginal expected revenue data transformation: A note on using old yield management techniques in the brave new world of pricing. J. Revenue Pricing Manag. **9**, 94–109 (2010)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.