On the Approximate Linear Programming Approach for Network Revenue Management Problems

Chaoxu Tong
School of Operations Research and Information Engineering,
Cornell University, Ithaca, New York 14853, USA
ct423@cornell.edu

Huseyin Topaloglu
School of Operations Research and Information Engineering,
Cornell University, Ithaca, New York 14853, USA
topaloglu@orie.cornell.edu

June 7, 2012

Abstract

One method to obtain high-quality bid prices for network revenue management problems involves using the approximate linear programming approach on the dynamic programming formulation of the problem. This approach ends up with a linear program whose number of constraints increases exponentially with the number of flight legs in the airline network. The linear program is solved by using constraint generation, where each constraint can be generated by solving a separate integer program. The necessity to solve integer programs and the slow convergence behavior of constraint generation are generally recognized as drawbacks of this approach. In this paper, we show how to effectively eliminate these drawbacks. In particular, we establish that constraint generation can actually be carried out by solving minimum-cost network flow problems with natural integer solutions. Furthermore, using the structure of minimum-cost network flow problems, we a priori reduce the number of constraints in the linear program from exponential in the number of flight legs to linear. It turns out that the reduced linear program can be solved without using separate problems to generate constraints. The reduced linear program also provides a practically appealing interpretation. Computational experiments indicate that our results can speed up the computation time for the approximate linear programming approach by a factor ranging between 13 and 135.

1 Introduction

Bid prices form a powerful tool to obtain high-quality control policies for network revenue management problems. The fundamental idea is to associate a bid price with each flight leg in the airline network, characterizing the opportunity cost of a seat on the flight leg. In this case, an itinerary is opened for sale to customers if the revenue that would be obtained from the sale of the itinerary exceeds the total opportunity cost of the seats consumed by the itinerary; see Williamson (1992) and Talluri and van Ryzin (1998). It is known that bid price policies are not necessarily optimal, but their ease of implementation and intuitive appeal make them a popular choice in practice.

Since the bid price characterizes the opportunity cost of a seat on a flight leg, it is natural to expect that the bid price should depend on how much time is left until the departure time. As the departure time of a flight leg approaches, there are fewer opportunities to utilize the capacities and one would intuitively expect that the bid price of a flight leg would decrease, all else being equal. Despite this intuitive expectation, numerous popular models in the literature provide static bid prices that do not depend on how much time is left until departure; see, for example, Talluri and van Ryzin (1998) and Talluri and van Ryzin (1999). As the sales take place and the departure time approaches, one has to periodically resolve these models to get bid prices that actually change as a function of the remaining time until departure. Recently, Adelman (2007) proposes a model that naturally provides bid prices that depend on time until departure. This model starts with the dynamic programming formulation of the network revenue management problem, which is difficult to solve since the state variable in this formulation is a high-dimensional vector. The idea is to approximate the value functions with affine functions and choose the slopes and intercepts of the affine approximations by plugging them into the linear programming representation of the dynamic program. The number of decision variables in this linear program is manageable, but the number of constraints grows exponentially with the number of flight legs. Therefore, constraint generation is a natural method for solving the linear program. The slopes of the affine value function approximations are ultimately used as bid prices.

This model is quite appealing as it has theoretical footing in dynamic programming theory and generates bid prices that depend on time left until departure, but its implementation in practice can be difficult. To begin with, solving the linear program by using constraint generation requires iteratively generating constraints and each constraint can be generated by solving a separate integer program. This integer program is a drawback when applying the model to large airline networks. Furthermore, constraint generation methods are known to be slow in obtaining the optimal solution; see Ruszczynski (2006). Finally, implementing constraint generation requires a good deal of customized software development and this negatively affects the practical appeal of the model.

In this paper, we consider the network revenue management model proposed by Adelman (2007) and eliminate essentially all of the drawbacks that hinder its practical implementation. First, we show that the integer program that needs to be solved for generating constraints can actually be formulated as a minimum-cost network flow problem whose continuous relaxation has no integrality gap. This result holds for an arbitrary airline network topology, allowing us to generate constraints very quickly even for large and complicated airline networks. The minimum-cost network flow formulation of the constraint

generation problem is of interest by itself, but we note that constraint generation may still be slow to ultimately obtain the optimal solution, although we can generate each constraint quickly. Second, the number of constraints in the linear program used by Adelman (2007) grows exponentially with the number of flight legs in the airline network. By using the minimum-cost network flow structure of the constraint generation problem, we a priori reduce this linear program to an equivalent linear program whose size grows only linearly with the numbers of flight legs and itineraries. The reduced linear program completely eliminates the need to solve separate problems to generate constraints. Third, we give an appealing intuitive interpretation for the reduced linear program that clearly shows its relationship to the network revenue management problem we want to solve. This interpretation is likely to enhance the appeal of the model to practitioners. Fourth, our computational experiments indicate that by using the results in this paper, we can solve the model proposed by Adelman (2007) up to 135 times faster. Over all of our test problems, the average speed up factor is about 52.

There are a number of models in the literature that are used to compute bid prices. Simpson (1989) and Williamson (1992) compute static bid prices by using a deterministic approximation to the network revenue management problem that is formulated under the assumption that all itinerary requests take on their expected values. In the deterministic approximation, there is a capacity constraint for each flight leg, ensuring that the served itinerary requests do not violate the leg capacities. The optimal values of the dual variables associated with these capacity constraints are used as bid prices. Talluri and van Ryzin (1998) give an analysis of the bid prices obtained by this deterministic approximation. Talluri and van Ryzin (1999) introduce randomness into the deterministic approximation by using samples of the itinerary requests, rather than expected values. Farias and van Roy (2007) and Meissner and Strauss (2008) use piecewise linear approximations to the value functions in the dynamic programming formulation of the problem and the policies obtained from their approaches can be viewed as bid price policies. Zhang and Adelman (2009) extend the model proposed by Adelman (2007) to cover the case where each customer makes a choice among the itineraries that are open for sale. Topaloglu (2009) computes bid prices that depend on both how much time is left until departure and how much capacity is left on the flight legs, but his approach is more computationally intensive than that of Adelman (2007). Kunnumkal and Topaloglu (2010) use Lagrangian relaxation to relax the capacity constraints in the dynamic programming formulation. The relaxed dynamic program can be solved efficiently and ultimately yields bid prices that depend on time.

The rest of the paper is organized as follows. In Section 2, we begin by formulating the network revenue management problem as a dynamic program. In Section 3, we give the linear programming representation of this dynamic program. To obtain tractable approximations to the value functions, we replace the value functions in the linear programming representation by affine approximations. In this case, we obtain a linear programming representation with manageable number of decision variables, but the number of constraints grows exponentially with the number of flight legs. In Section 4, we describe how to solve the linear programming representation by using constraint generation and show that each constraint can be generated by solving a minimum-cost network flow problem. In Section 5, we exploit the minimum-cost network flow structure to a priori reduce the linear programming representation into a linear program whose numbers of constraints and decision variables grow only

linearly with the numbers of flight legs and itineraries. In Section 6, we give a practical interpretation of this reduced linear program. In Section 7, we present computational experiments that demonstrate the computational benefits from the reduced linear program. In Section 8, we conclude.

2 Problem Formulation

We have a set of flight legs over an airline network that can be used to serve the requests for itineraries that arrive randomly over time. At the beginning of each time period, we decide whether each itinerary is open for sale or closed. If there is a request for an itinerary that is open for sale, then we serve this itinerary request, generating a revenue and consuming capacities on the flight legs that are included in the requested itinerary. A request for a closed itinerary simply leaves the system. The objective is to find a policy to open the itineraries for sale or close them over time so as to maximize the total expected revenue from the served itinerary requests.

We let $\mathcal{L}$ be the set of flight legs in the airline network and $\mathcal{J}$ be the set of itineraries. If we serve a request for itinerary j, then we generate a revenue of f_j and consume a_{ij} units of capacity on flight leg i. We assume that there are no group reservations so that each served itinerary request consumes at most one unit of capacity on a flight leg. In other words, we have $a_{ij} \in \{0,1\}$ for all $i \in \mathcal{L}$, $j \in \mathcal{J}$. Thus, an itinerary j is characterized by the set of flight legs $\{i \in \mathcal{L} : a_{ij} = 1\}$ that it uses and the revenue f_j that it generates. In certain revenue management settings, an itinerary is referred to as a product. The available capacity on flight leg i is c_i. The itinerary requests arrive one by one over the time periods $\mathcal{T} = \{1, \ldots, \tau\}$. Arrivals of the itinerary requests at different time periods are independent. The probability that there is a request for itinerary j at time period t is p_{jt}. We assume that $\sum_{j \in \mathcal{J}} p_{jt} = 1$ so that there is exactly one itinerary request at each time period. If there is a positive probability of having no itinerary request at time period t, then we can capture this situation by defining a dummy itinerary ϕ satisfying $f_\phi = 0$, $a_{i\phi} = 0$ for all $i \in \mathcal{L}$ and $p_{\phi t} = 1 - \sum_{j \in \mathcal{J}} p_{jt}$.

We use x_{it} to denote the remaining capacity on flight leg i at the beginning of time period t so that the vector $x_t = \{x_{it} : i \in \mathcal{L}\}$ captures the state of the capacities on the flight legs at the beginning of this time period. We let u_{jt} take value one if we open itinerary j for sale at time period t and take value zero otherwise. The vector $u_t = \{u_{jt} : j \in \mathcal{J}\}$ captures the decisions at time period t. The set of feasible decisions is given by $\mathcal{U}(x_t) = \{u_t \in \{0,1\}^{|\mathcal{J}|} : a_{ij}\, u_{jt} \leq x_{it}\ \forall\, i \in \mathcal{L},\ j \in \mathcal{J}\}$, ensuring that if we want to open itinerary j for sale and itinerary j uses flight leg i, then there has to be capacity available on flight leg i. In this case, we can formulate the problem as a dynamic program as

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

where we use e_i to denote the $|\mathcal{L}|$-dimensional unit vector with a one in the element corresponding to flight leg i. The boundary condition of the optimality equation above is $V_{\tau+1}(\cdot) = 0$ and the state space can be written as $\mathcal{X} = \prod_{i \in \mathcal{L}} \{0, 1, \ldots, c_i\}$. Assuming that we have access to the value functions $\{V_t(x_t) : x_t \in \mathcal{X},\ t \in \mathcal{T}\}$, it is not difficult to compute the optimal decisions at each time period. If we have $a_{ij} \leq x_{it}$ for all $i \in \mathcal{L}$ and $f_j + V_{t+1}(x_t - \sum_{i \in \mathcal{L}} a_{ij}\, e_i) \geq V_{t+1}(x_t)$, then it is optimal to open

itinerary j for sale at time period t when the remaining capacities on the flight legs are given by the vector x_t. Otherwise, it is optimal to close itinerary j.

The size of the state space $\mathcal{X}$ in the optimality equation in (1) grows exponentially with the number of flight legs, rendering the computation of the value functions intractable for essentially any practical airline network. In this paper, we focus on an approximate solution method proposed by Adelman (2007). We describe this solution method in the next section.

3 Approximate Linear Program

It is a standard result in Markov decision processes that an optimality equation with finite sets of states and decisions can be formulated as a linear program; see Puterman (1994). In general, this linear program is not computationally useful by itself since its numbers of decision variables and constraints are proportional to the number of possible states, but the linear program can serve as a starting point for constructing approximations to the value functions. Using c to denote the vector of initial capacities $\{c_i : i \in \mathcal{L}\}$, the linear program corresponding to the optimality equation in (1) can be written as

$$\begin{aligned} \min \quad & \vartheta_1(c) && \text{(2a)} \\ \text{subject to} \quad & \vartheta_t(x_t) \geq \sum_{j \in \mathcal{J}} p_{jt} \Big\{ f_j \, u_{jt} + \vartheta_{t+1}(x_t - u_{jt} \textstyle\sum_{i \in \mathcal{L}} a_{ij} \, e_i) \Big\} \\ & && \forall \, x_t \in \mathcal{X}, \; u_t \in \mathcal{U}(x_t), \; t \in \mathcal{T}, && \text{(2b)} \end{aligned}$$

where the decision variables are $\{\vartheta_t(x_t) : x_t \in \mathcal{X}, \; t \in \mathcal{T}\}$. For notational uniformity, we follow the convention that the values of the decision variables $\{\vartheta_{\tau+1}(x_{\tau+1}) : x_{\tau+1} \in \mathcal{X}\}$ in the problem above are set to zero. It is possible to show that the optimal objective value of problem (2) is equal to the optimal total expected revenue $V_1(c)$ that is obtained through the optimality equation in (1). Therefore, the objective function of problem (2) evaluated at any feasible solution provides an upper bound on the optimal total expected revenue.

Problem (2) is not computationally useful by itself since the numbers of decision variables and constraints in this problem increase exponentially with the number of flight legs. To deal with this difficulty, Adelman (2007) proposes approximating the value function $V_t(x_t)$ with an affine function of the form $\tilde{V}_t(x_t) = \theta_t + \sum_{i \in \mathcal{L}} v_{it} \, x_{it}$ for all $t \in \mathcal{T}$, where $\{\theta_t : t \in \mathcal{T}\}$ and $\{v_{it} : i \in \mathcal{L}, \; t \in \mathcal{T}\}$ are adjustable parameters. To choose a set of values for $\{\theta_t : t \in \mathcal{T}\}$ and $\{v_{it} : i \in \mathcal{L}, \; t \in \mathcal{T}\}$, we plug the approximation $\tilde{V}_t(x_t) = \theta_t + \sum_{i \in \mathcal{L}} v_{it} \, x_{it}$ into problem (2) to obtain the linear program

$$\begin{aligned} \min \quad & \theta_1 + \sum_{i \in \mathcal{L}} v_{i1} \, c_i && \text{(3a)} \\ \text{subject to} \quad & \theta_t + \sum_{i \in \mathcal{L}} v_{it} \, x_{it} \geq \sum_{j \in \mathcal{J}} p_{jt} \Big\{ f_j \, u_{jt} + \theta_{t+1} + \sum_{i \in \mathcal{L}} v_{i,t+1} \, [x_{it} - u_{jt} \, a_{ij}] \Big\} \\ & && \forall \, x_t \in \mathcal{X}, \; u_t \in \mathcal{U}(x_t), \; t \in \mathcal{T}, && \text{(3b)} \end{aligned}$$

where the decision variables are $\{\theta_t : t \in \mathcal{T}\}$ and $\{v_{it} : i \in \mathcal{L}, \; t \in \mathcal{T}\}$. Similar to problem (2), we follow the convention that the values of the decision variables $\theta_{\tau+1}$ and $\{v_{i,\tau+1} : i \in \mathcal{L}\}$ in the problem above

are set to zero. The number of constraints in problem (3) still increases exponentially with the number of flight legs, but the number of decision variables is $|\mathcal{T}| + |\mathcal{L}|\,|\mathcal{T}|$, which can be manageable for airline networks of practical significance. Therefore, it can be possible to solve problem (3) by using constraint generation for practical airline networks.

There are two uses of problem (3). First, it is possible to show that the optimal objective value of this problem provides an upper bound on the optimal total expected revenue. In particular, letting $\{\theta_t^* : t \in \mathcal{T}\}$ and $\{v_{it}^* : i \in \mathcal{L},\ t \in \mathcal{T}\}$ be an optimal solution to problem (3) and $\hat{\vartheta}_t(x_t) = \theta_t^* + \sum_{i \in \mathcal{L}} v_{it}^*\, x_{it}$ for all $x_t \in \mathcal{X}$, $t \in \mathcal{T}$, we observe that $\{\hat{\vartheta}_t(x_t) : x_t \in \mathcal{X},\ t \in \mathcal{T}\}$ is a feasible solution to problem (2). Since the objective function of problem (2) evaluated at a feasible solution provides an upper bound on the optimal total expected revenue, it immediately follows that $\hat{\vartheta}_1(c) = \theta_1^* + \sum_{i \in \mathcal{L}} v_{i1}^*\, c_i$ is an upper bound on the optimal total expected revenue. Such an upper bound on the optimal total expected revenue becomes useful when we try to assess the optimality gap of approximate or heuristic control policies. Second, we can use an optimal solution to problem (3) to construct an approximate control policy. Recalling that $\{\theta_t^* : t \in \mathcal{T}\}$ and $\{v_{it}^* : i \in \mathcal{L},\ t \in \mathcal{T}\}$ denote an optimal solution to problem (3), the idea is to use $\tilde{V}_t^*(x_t) = \theta_t^* + \sum_{i \in \mathcal{L}} v_{it}^*\, x_{it}$ as an approximation to $V_t(x_t)$. In this case, if we have $a_{ij} \leq x_{it}$ for all $i \in \mathcal{L}$ and $f_j + \theta_{t+1}^* + \sum_{i \in \mathcal{L}} v_{i,t+1}^*\,[x_{it} - a_{ij}] \geq \theta_{t+1}^* + \sum_{i \in \mathcal{L}} v_{i,t+1}^*\, x_{it}$, then we open itinerary j for sale at time period t when the remaining capacities on the flight legs are given by the vector x_t. To gain some insight into this policy, we write the last inequality as

$$f_j \geq \sum_{i \in \mathcal{L}} a_{ij}\, v_{i,t+1}^*.$$

We interpret v_{it}^* in the value function approximation $\tilde{V}_t^*(x_t) = \theta_t^* + \sum_{i \in \mathcal{L}} v_{it}^*\, x_{it}$ as the marginal value or the opportunity cost of a unit of capacity on flight leg i at time period t, in which case, the right side of the inequality above corresponds to the total value of the capacities used by itinerary j. Therefore, the control policy obtained from problem (3) compares the revenue from itinerary j with the total value of the capacities consumed by itinerary j and it opens itinerary j for sale if the revenue justifies the total value of the consumed capacities, as long as there are enough seats.

In network revenue management, the marginal value or the opportunity cost of a seat is referred to as the bid price. Intuitively, the bid price of a flight leg, being the marginal value of a unit of capacity, should depend on how much time is left until the time of departure. An attractive feature of problem (3) is that it naturally provides bid prices that are indeed dependent on time. Throughout the rest of the paper, we focus on efficient solution methods for problem (3).

4 Constraint Generation

Noting that the number of decision variables in problem (3) is manageable but the number of constraints grows exponentially with the number of flight legs, a possible solution method for this problem is to use constraint generation. The idea behind constraint generation is to iteratively solve a master problem that has the same decision variables as problem (3), but has only a subset of the constraints. After solving the master problem, we check whether any one of the constraints in (3b) is violated by the solution to the current master problem. If there is one such constraint, then we add this constraint

to the master problem and resolve the master problem. Otherwise, the solution to the current master problem is optimal to problem (3) and we stop.

The key to efficient implementation of constraint generation is to be able to check whether the solution to the current master problem violates any one of the constraints in (3b). Using the fact that $\sum_{j \in \mathcal{J}} p_{jt} = 1$, we write constraints (3b) as

$$\theta_t - \theta_{t+1} \geq \sum_{j \in \mathcal{J}} p_{jt} \left\{ f_j - \sum_{i \in \mathcal{L}} a_{ij} \, v_{i,t+1} \right\} u_{jt} + \sum_{i \in \mathcal{L}} \left[v_{i,t+1} - v_{it} \right] x_{it}$$

for all $x_t \in \mathcal{X}$, $u_t \in \mathcal{U}(x_t)$, $t \in \mathcal{T}$. In this case, if we let $\{\tilde{\theta}_t : t \in \mathcal{T}\}$ and $\{\tilde{v}_{it} : i \in \mathcal{L}, \ t \in \mathcal{T}\}$ be the solution to the current master problem, then we can check whether this solution violates any one of the constraints in (3b) by solving

$$\max_{x_t \in \mathcal{X}, \ u_t \in \mathcal{U}(x_t)} \left\{ \sum_{j \in \mathcal{J}} p_{jt} \left\{ f_j - \sum_{i \in \mathcal{L}} a_{ij} \, \tilde{v}_{i,t+1} \right\} u_{jt} + \sum_{i \in \mathcal{L}} \left[\tilde{v}_{i,t+1} - \tilde{v}_{it} \right] x_{it} \right\} \tag{4}$$

for all $t \in \mathcal{T}$. If the optimal objective value of the problem above exceeds $\tilde{\theta}_t - \tilde{\theta}_{t+1}$ for a particular time period t, then letting $\tilde{x}_t$ and $\tilde{u}_t$ be the optimal solution to problem (4), the constraint corresponding to $\tilde{x}_t \in \mathcal{X}$, $\tilde{u}_t \in \mathcal{U}(\tilde{x}_t)$ and $t \in \mathcal{T}$ in problem (3) is violated by the solution to the current master problem. We add this constraint to the current master problem and solve the master problem with the added constraint. Therefore, efficient implementation of constraint generation is dependent on our ability to solve problem (4) quickly.

Noting the definitions of the state space $\mathcal{X}$ and the set of feasible decisions $\mathcal{U}(x_t)$, the decision variables x_t and u_t in problem (4) have integrality constraints. Adelman (2007) notes that we can relax the integrality constraints on the decision variables x_t without loss of optimality, but we still have the integrality constraints on the decision variables u_t in problem (4). Due to such integrality constraints, problem (4) is recognized as a drawback when using constraint generation to solve problem (3) for large airline networks. In this section, we alleviate such concerns by showing that the continuous relaxation of problem (4) has no integrality gap. Therefore, we can generate constraints by solving the continuous relaxation of problem (4), which can be done efficiently for practical airline networks.

To show that the continuous relaxation of problem (4) has no integrality gap, we exploit the fact that the dual of this continuous relaxation turns out to be a minimum-cost network flow problem with integer cost data. Using the definitions of the state space $\mathcal{X}$ and the set of feasible decisions $\mathcal{U}(x_t)$, we write the continuous relaxation of problem (4) as

$$\begin{aligned}
\max \quad & \sum_{j \in \mathcal{J}} p_{jt} \left\{ f_j - \sum_{i \in \mathcal{L}} a_{ij} \, \tilde{v}_{i,t+1} \right\} u_{jt} + \sum_{i \in \mathcal{L}} \left[\tilde{v}_{i,t+1} - \tilde{v}_{it} \right] x_{it} && && \text{(5a)} \\
\text{subject to} \quad & x_{it} \leq c_i && \forall \, i \in \mathcal{L} && \text{(5b)} \\
& a_{ij} \, u_{jt} \leq x_{it} && \forall \, i \in \mathcal{L}, \ j \in \mathcal{J} && \text{(5c)} \\
& u_{jt} \leq 1 && \forall \, j \in \mathcal{J} && \text{(5d)} \\
& x_{it} \text{ is free}, \ u_{jt} \geq 0 && \forall \, i \in \mathcal{L}, \ j \in \mathcal{J}. && \text{(5e)}
\end{aligned}$$

We do not explicitly impose nonnegativity constraints on the decision variables $\{x_{it} : i \in \mathcal{L}\}$ in the problem above, since constraints (5c) along with the nonnegativity constraints on the decision variables $\{u_{jt} : j \in \mathcal{J}\}$ ensure that the decision variables $\{x_{it} : i \in \mathcal{L}\}$ are nonnegative. Our goal is to show that the dual of problem (5) is a minimum-cost network flow problem. To that end, we associate the dual variables $\{\alpha_{it} : i \in \mathcal{L}\}$, $\{\beta_{ijt} : i \in \mathcal{L},\ j \in \mathcal{J}\}$, and $\{\gamma_{jt} : j \in \mathcal{J}\}$ respectively with constraints (5b), (5c) and (5d) in problem (5). In the dual of problem (5), the constraints associated with the decision variables $\{u_{jt} : j \in \mathcal{J}\}$ are given by $\sum_{i \in \mathcal{L}} a_{ij}\, \beta_{ijt} + \gamma_{jt} \geq p_{jt}\, [f_j - \sum_{i \in \mathcal{L}} a_{ij}\, \tilde{v}_{i,t+1}]$ for all $j \in \mathcal{J}$. Using the slack variables $\{\mu_{jt} : j \in \mathcal{J}\}$ for these constraints, the dual of problem (5) is given by

$$\begin{aligned}
\min \quad & \sum_{i \in \mathcal{L}} c_i\, \alpha_{it} + \sum_{j \in \mathcal{J}} \gamma_{jt} && && \text{(6a)}\\
\text{subject to} \quad & \alpha_{it} - \sum_{j \in \mathcal{J}} \beta_{ijt} = \tilde{v}_{i,t+1} - \tilde{v}_{it} && \forall\, i \in \mathcal{L} && \text{(6b)}\\
& \sum_{i \in \mathcal{L}} a_{ij}\, \beta_{ijt} + \gamma_{jt} - \mu_{jt} = p_{jt} \left\{ f_j - \sum_{i \in \mathcal{L}} a_{ij}\, \tilde{v}_{i,t+1} \right\} && \forall\, j \in \mathcal{J} && \text{(6c)}\\
& \alpha_{it} \geq 0,\ \beta_{ijt} \geq 0,\ \gamma_{jt} \geq 0,\ \mu_{jt} \geq 0 && \forall\, i \in \mathcal{L},\ j \in \mathcal{J}. && \text{(6d)}
\end{aligned}$$

The next proposition shows that problem (6) is a minimum-cost network flow problem.

Proposition 1 *Problem* (6) *is a minimum-cost network flow problem.*

Proof We establish the result by showing that all of the decision variables in problem (6) can be associated with the arcs in a directed network and all of the constraints in problem (6) correspond to the flow balance constraints of the nodes in this directed network. We consider a network with two sets of nodes $\mathcal{N}_1 = \{i : i \in \mathcal{L}\}$ and $\mathcal{N}_2 = \{j : j \in \mathcal{J}\}$ and an additional sink node. Constraints (6b) and (6c) are respectively the flow balance constraints for the nodes in $\mathcal{N}_1$ and $\mathcal{N}_2$. The flow balance constraint for the sink node is redundant and it is omitted in problem (6). The decision variable α_{it} in problem (6) corresponds to an arc that connects the sink node to node $i \in \mathcal{N}_1$. If $a_{ij} = 1$, then the decision variable β_{ijt} corresponds to an arc from node $i \in \mathcal{N}_1$ to node $j \in \mathcal{N}_2$. On the other hand, if $a_{ij} = 0$, then the decision variable β_{ijt} corresponds to an arc from node $i \in \mathcal{N}_1$ to the sink node. Therefore, the decision variable β_{ijt} does not appear in the flow balance constraint for node $j \in \mathcal{N}_2$ when $a_{ij} = 0$. The decision variable γ_{jt} corresponds to an arc from the sink node to node $j \in \mathcal{N}_2$. The decision variable μ_{jt} corresponds to an arc from node $j \in \mathcal{N}_2$ to the sink node. The demands of nodes $i \in \mathcal{N}_1$ and $j \in \mathcal{N}_2$ are respectively $\tilde{v}_{i,t+1} - \tilde{v}_{it}$ and $p_{jt}\, [f_j - \sum_{i \in \mathcal{L}} a_{ij}\, \tilde{v}_{i,t+1}]$. □

Figure 1 shows the structure of problem (6) for a small airline network. The upper portion of this figure shows the airline network. The flight legs are labeled by $\{1, 2\}$ and represented in solid arcs, whereas the itineraries are labeled by $\{a, b, c\}$ and represented in dotted arcs. The lower portion of the figure shows the minimum-cost network flow problem corresponding to problem (6). Each arc in Figure 1 corresponds to a decision variable in problem (6) and each node with the exception of the sink node corresponds to a constraint.

Proposition 1 immediately implies that problem (5) has no integrality gap. In particular, problems (5) and (6) form a primal-dual pair so that an optimal dual solution to problem (6) is an optimal

solution to problem (5). Since problem (6) is a minimum-cost network flow problem with integer cost data, there exists an integer-valued optimal dual solution to problem (6) and this integer-valued optimal dual solution can be obtained by solving problem (6) by using the simplex algorithm; see Vanderbei (1997). Therefore, there exists an integer-valued optimal solution to problem (5), establishing that this problem has no integrality gap.

We do not make any assumptions in Proposition 1 on the structure of the underlying airline network, implying that problem (5) has no integrality gap for any airline network structure. Furthermore, we can solve the dual of problem (5) by using minimum-cost network flow algorithms, which are quite efficient. These observations indicate that constraint generation is efficient to implement even for large airline networks with arbitrary topologies. It is also worthwhile to observe that it is necessary to have $a_{ij} \in \{0,1\}$ for all $i \in \mathcal{L}$, $j \in \mathcal{J}$ for Proposition 1 to hold. Otherwise, constraints (6c) cannot be interpreted as flow balance constraints.

The fact that problem (5) has no integrality gap is of interest by itself, but we also build on this result in the next section to show that we can reduce problem (3) to a linear program whose size grows linearly with the numbers of flight legs and itineraries.

5 Reducing the Number of Constraints

By the discussion in the previous section, if we use constraint generation to solve problem (3), then we can generate each constraint efficiently. However, although generating each constraint is efficient, we may still end up generating a large number of constraints to ultimately obtain the optimal solution by using constraint generation. In this section, we establish a complementary result, showing that we can a priori reduce problem (3) to a linear program whose numbers of constraints and decision variables grow only linearly with the numbers of flight legs and itineraries. This result significantly enhances the computational tractability of problem (3).

To facilitate our discussion, we use v to denote the collection of decision variables $\{v_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ in problem (3) and define $\Pi_t(v)$ as

$$\Pi_t(v) = \max_{x_t \in \mathcal{X},\ u_t \in \mathcal{U}(x_t)} \left\{ \sum_{j \in \mathcal{J}} p_{jt} \left\{ f_j - \sum_{i \in \mathcal{L}} a_{ij}\, v_{i,t+1} \right\} u_{jt} + \sum_{i \in \mathcal{L}} \left[v_{i,t+1} - v_{it} \right] x_{it} \right\}. \tag{7}$$

The notation $\Pi_t(v)$ suggests that the quantity on the right side above depends on the whole collection $v = \{v_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$, but $\Pi_t(v)$ actually depends only on v_t and v_{t+1}. Our choice of notation for $\Pi_t(v)$ is motivated by notational brevity. With this definition of $\Pi_t(v)$, constraints (3b) in problem (3) can succinctly be written as $\theta_t - \theta_{t+1} \geq \Pi_t(v)$ for all $t \in \mathcal{T}$ and we can write problem (3) as

$$\min \quad \theta_1 + \sum_{i \in \mathcal{L}} v_{i1}\, c_i \tag{8a}$$

$$\text{subject to} \quad \theta_t - \theta_{t+1} \geq \Pi_t(v) \qquad \forall\, t \in \mathcal{T}, \tag{8b}$$

where the decision variables are $\{\theta_t : t \in \mathcal{T}\}$ and $\{v_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$. We make two useful observations in problem (8). First, the values of the decision variables $\{\theta_t : t \in \mathcal{T}\}$ and $\{v_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ have to

satisfy $\theta_t = \theta_{t+1} + \Pi_t(v)$ for all $t \in \mathcal{T}$ in an optimal solution to problem (8). Adding the last equality over all $t \in \mathcal{T}$ and noting the convention that the value of the decision variable $\theta_{\tau+1}$ is set to zero, it follows that $\theta_1 = \sum_{t \in \mathcal{T}} \Pi_t(v)$ in an optimal solution to problem (8). Second, we observe that problems (4) and (7) have the same structure. Therefore, Proposition 1 implies that the continuous relaxation of problem (7) has no integrality gap. By using the same argument that we use just before Proposition 1, we write the dual of the continuous relaxation of problem (7) as

$$\begin{aligned}
\Pi_t(v) = \min \quad & \sum_{i \in \mathcal{L}} c_i \, \alpha_{it} + \sum_{j \in \mathcal{J}} \gamma_{jt} && && \text{(9a)}\\
\text{subject to} \quad & \alpha_{it} - \sum_{j \in \mathcal{J}} \beta_{ijt} = v_{i,t+1} - v_{it} && \forall \, i \in \mathcal{L} && \text{(9b)}\\
& \sum_{i \in \mathcal{L}} a_{ij} \, \beta_{ijt} + \gamma_{jt} \geq p_{jt} \Big\{ f_j - \sum_{i \in \mathcal{L}} a_{ij} \, v_{i,t+1} \Big\} && \forall \, j \in \mathcal{J} && \text{(9c)}\\
& \alpha_{it} \geq 0, \ \beta_{ijt} \geq 0, \ \gamma_{jt} \geq 0 && \forall \, i \in \mathcal{L}, \ j \in \mathcal{J}, && \text{(9d)}
\end{aligned}$$

in which case, the problem above and problem (7) share the same optimal objective value and this common optimal objective value is denoted by $\Pi_t(v)$.

In the next proposition, we build on the two observations above to reduce problem (3) to a linear program whose numbers of constraints and decision variables grow linearly with the numbers of flight legs and itineraries.

Proposition 2 *Problem* (3) *is equivalent to the problem*

$$\begin{aligned}
\min \quad & \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{L}} c_i \, \alpha_{it} + \sum_{t \in \mathcal{T}} \sum_{j \in \mathcal{J}} \gamma_{jt} + \sum_{i \in \mathcal{L}} c_i \, v_{i1} && && \text{(10a)}\\
\text{subject to} \quad & \alpha_{it} - \sum_{j \in \mathcal{J}} \beta_{ijt} = v_{i,t+1} - v_{it} && \forall \, i \in \mathcal{L}, \ t \in \mathcal{T} && \text{(10b)}\\
& \sum_{i \in \mathcal{L}} a_{ij} \, \beta_{ijt} + \gamma_{jt} \geq p_{jt} \Big\{ f_j - \sum_{i \in \mathcal{L}} a_{ij} \, v_{i,t+1} \Big\} && \forall \, j \in \mathcal{J}, \ t \in \mathcal{T} && \text{(10c)}\\
& \alpha_{it} \geq 0, \ \beta_{ijt} \geq 0, \ \gamma_{jt} \geq 0, \ v_{it} \text{ is free} && \forall \, i \in \mathcal{L}, \ j \in \mathcal{J}, \ t \in \mathcal{T}. && \text{(10d)}
\end{aligned}$$

In particular, problems (3) *and* (10) *have the same optimal objective value and given an optimal solution to one problem, we can construct an optimal solution to the other.*

Proof Since problems (3) and (8) are equivalent to each other, we show that problem (8) is equivalent to problem (10). We use Z^* and ζ^* to respectively denote the optimal objective values of problems (8) and (10). First, we show that $Z^* \leq \zeta^*$. Since the optimal objective value of problem (9) gives $\Pi_t(v)$, problem (10) is equivalent to minimizing $\sum_{t \in \mathcal{T}} \Pi_t(v) + \sum_{i \in \mathcal{L}} c_i \, v_{i1}$ over v, which implies that $\zeta^* = \min_v \{ \sum_{t \in \mathcal{T}} \Pi_t(v) + \sum_{i \in \mathcal{L}} c_i \, v_{i1} \}$. So, letting v^* be the optimal solution to the last optimization problem and defining θ_t^* as $\theta_t^* = \sum_{s=t}^{\tau} \Pi_s(v^*)$, the solution $\{\theta_t^* : t \in \mathcal{T}\}$ and $v^* = \{v_{it}^* : i \in \mathcal{L}, \ t \in \mathcal{T}\}$ is feasible to problem (8) and we obtain $Z^* \leq \sum_{t \in \mathcal{T}} \Pi_t(v^*) + \sum_{i \in \mathcal{L}} c_i \, v_{i1}^* = \zeta^*$.

Second, we show that $\zeta^* \leq Z^*$. We let $\{\theta_t^* : t \in \mathcal{T}\}$ and $v^* = \{v_{it}^* : i \in \mathcal{L}, \ t \in \mathcal{T}\}$ be an optimal solution to problem (8). To construct a feasible solution to problem (10), we solve problem

(9) for each $t \in \mathcal{T}$ after replacing the right side of constraints (9b) with $\{v^*_{i,t+1} - v^*_{it} : i \in \mathcal{L}\}$ and the right side of constraints (9c) with $\{p_{jt}\,[f_j - \sum_{i\in\mathcal{L}} a_{ij}\, v^*_{i,t+1}] : j \in \mathcal{J}\}$. Letting $\{\alpha^*_{it} : i \in \mathcal{L}\}$, $\{\beta^*_{ijt} : i \in \mathcal{L},\ j \in \mathcal{J}\}$ and $\{\gamma^*_{jt} : j \in \mathcal{J}\}$ be an optimal solution to problem (9) for each $t \in \mathcal{T}$, we have $\Pi_t(v^*) = \sum_{i\in\mathcal{L}} c_i\, \alpha^*_{it} + \sum_{j\in\mathcal{J}} \gamma^*_{jt}$ for all $t \in \mathcal{T}$. Also, noting constraints (9b) and (9c), the solution $\{\alpha^*_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$, $\{\beta^*_{ijt} : i \in \mathcal{L},\ j \in \mathcal{J}\ , t \in \mathcal{T}\}$, $\{\gamma^*_{jt} : j \in \mathcal{J},\ t \in \mathcal{T}\}$ and $\{v^*_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ is feasible to problem (10), but not necessarily optimal. Therefore, we obtain

$$\zeta^* \leq \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{L}} c_i\, \alpha^*_{it} + \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} \gamma^*_{jt} + \sum_{i\in\mathcal{L}} c_i\, v^*_{i1} = \sum_{t\in\mathcal{T}} \Pi_t(v^*) + \sum_{i\in\mathcal{L}} c_i\, v^*_{i1} = \theta^*_1 + \sum_{i\in\mathcal{L}} c_i\, v^*_{i1} = Z^*,$$

where the second equality uses the fact $\theta^*_1 = \sum_{t\in\mathcal{T}} \Pi_t(v^*)$ is always satisfied by an optimal solution to problem (8). Thus, we obtain $Z^* = \zeta^*$ and we can use the construction above to get an optimal solution to one of the problems (3) and (10) by using an optimal solution to the other. □

The proof of Proposition 2 shows that if $\{\alpha^*_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$, $\{\beta^*_{ijt} : i \in \mathcal{L},\ j \in \mathcal{J},\ t \in \mathcal{T}\}$, $\{\gamma^*_{jt} : j \in \mathcal{J},\ t \in \mathcal{T}\}$ and $v^* = \{v^*_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ is an optimal solution to problem (10), then letting $\theta^*_t = \sum_{s=t}^{\tau} \Pi_s(v^*)$, the solution $\{\theta^*_t : t \in \mathcal{T}\}$ and $\{v^*_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ is optimal to problem (3). Therefore, we can recover an optimal solution to problem (3) from problem (10).

6 Practical Interpretation

At first glance, it is difficult to see an intuitive relationship between problem (10) and the network revenue management problem that we are interested in solving. In this section, we give a practical interpretation for problem (10) that clarifies this relationship. This practical interpretation also becomes useful to construct an efficient solution method for problem (10) later in this section.

Associating the dual variables $\{w_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ and $\{y_{jt} : j \in \mathcal{J},\ t \in \mathcal{T}\}$ respectively with constraints (10b) and (10c) in problem (10), we write the dual of this problem as

$$\begin{aligned}
\max \quad & \sum_{t\in\mathcal{T}} \sum_{j\in\mathcal{J}} p_{jt}\, f_j\, y_{jt} && && (11a)\\
\text{subject to} \quad & w_{i1} = c_i && \forall\, i \in \mathcal{L} && (11b)\\
& w_{it} = w_{i,t-1} - \sum_{j\in\mathcal{J}} p_{j,t-1}\, a_{ij}\, y_{j,t-1} && \forall\, i \in \mathcal{L},\ t \in \mathcal{T} \setminus \{1\} && (11c)\\
& a_{ij}\, y_{jt} \leq w_{it} && \forall\, i \in \mathcal{L},\ j \in \mathcal{J},\ t \in \mathcal{T} && (11d)\\
& y_{jt} \leq 1 && \forall\, j \in \mathcal{J},\ t \in \mathcal{T} && (11e)\\
& w_{it} \text{ is free},\ y_{jt} \geq 0 && \forall\, i \in \mathcal{L},\ j \in \mathcal{J},\ t \in \mathcal{T}. && (11f)
\end{aligned}$$

Constraints (11b) and (11c) are associated with the decision variables $\{v_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ in problem (10), whereas the constraints (11d) and (11e) are respectively associated with the decision variables $\{\beta_{ijt} : i \in \mathcal{L},\ j \in \mathcal{J},\ t \in \mathcal{T}\}$ and $\{\gamma_{jt} : j \in \mathcal{J},\ t \in \mathcal{T}\}$. We observe that the constraints associated with the decision variables $\{\alpha_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ in the dual of problem (10) are given by $w_{it} \leq c_i$ for all $i \in \mathcal{L}$, $t \in \mathcal{T}$, but constraints (11b) and (11c) already imply that $c_i = w_{i1} \geq w_{i2} \geq \ldots \geq w_{i\tau}$ for all $i \in \mathcal{L}$. Therefore, the constraints associated with the decision variables $\{\alpha_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$

are redundant in problem (11) and they are omitted. We observe that the size of the problem above increases linearly with the numbers of flight legs and itineraries.

We view the decision variable w_{it} in problem (11) as the expected remaining capacity on flight leg i at the beginning of time period t and the decision variable y_{jt} as the probability with which we open itinerary j for sale at time period t. At time period t, we open each itinerary j for sale with probability y_{jt}, independent of the other itineraries. With probability $1 - y_{jt}$, itinerary j is closed at time period t. To make a sale for itinerary j at time period t, we need to have itinerary j open and have a request for this itinerary, which happens with probability $p_{jt}\, y_{jt}$. Thus, the objective function of problem (11) accounts for the total expected revenue from the served itinerary requests. Constraints (11b) initialize the expected remaining capacities. Noting that $\sum_{j \in \mathcal{J}} p_{jt}\, a_{ij}\, y_{jt}$ is the expected capacity consumption on flight leg i at time period t, constraints (11c) compute the expected remaining capacities at the next time period as a function of the expected remaining capacities and itinerary sales at the current time period. Constraints (11d) ensure that the expected capacity consumption on flight leg i at time period t conditional on the fact that there is a request for itinerary j does not exceed the expected remaining capacity on flight leg i.

The number of constraints in problem (11) is substantially smaller than the number of constraints in problem (3), but problem (11) still has $|\mathcal{L}||\mathcal{T}| + |\mathcal{L}||\mathcal{J}||\mathcal{T}|$ constraints, which may be too many to solve this problem directly by using a linear programming solver. Noting that constraints (11d) can be replaced by nonnegativity constraints on the decision variables $\{w_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ whenever itinerary j does not use flight leg i, if we let L be the maximum number of flight legs that an itinerary uses, then we can reduce the number of constraints to $|\mathcal{L}||\mathcal{T}| + L|\mathcal{J}||\mathcal{T}|$, but this may also be too many. The key observation that allows us to solve problem (11) efficiently is that the expected remaining capacities $\{w_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ are initialized to the total available capacities by constraints (11b). Since $a_{ij} \in \{0, 1\}$ and y_{jt} is bounded by one, constraints (11d) are not active whenever w_{it} exceeds one, which indicates that we do not expect constraints (11d) to be tight over a large portion of the selling horizon. This observation motivates solving problem (11) by using constraint generation, where we iteratively solve a master problem that has the same decision variables as problem (11) and includes all of constraints (11b), (11c) and (11e), but has only a subset of constraints (11d). After solving the master problem, we check whether any one of the constraints in (11d) are violated by the solution to the current master problem. If there is one, then we add this constraint to the master problem and resolve it.

When compared with solving problem (3) by using constraint generation, applying constraint generation on problem (11) provides two potential advantages. First, we can simply enumerate over all $i \in \mathcal{L}$, $j \in \mathcal{J}$ and $t \in \mathcal{T}$ to check whether any one of the constraints in (11d) are violated by the solution to the current master problem. We do not need to solve a separate problem to identify violated constraints. Second, we may be interested in solving problem (11) not only to obtain a control policy, but also to obtain the upper bound on the optimal total expected revenue provided by the optimal objective value of this problem. In practice, it may not be possible to solve problem (11) to optimality by using constraint generation. However, since problem (11) is a maximization problem and the master problem includes a subset of its constraints, the optimal objective value of the master problem is always an upper

bound on the optimal objective value of problem (11). Therefore, if we stop constraint generation at an intermediate iteration, then the optimal objective value of the master problem naturally provides an upper bound on the optimal total expected revenue. This observation does not immediately apply when we solve problem (3) by using constraint generation since this problem is a minimization problem, but as Proposition 3 in Adelman (2007) shows, we can use the slack variables of the constraints in problem (3) to obtain an upper bound on the optimal total expected revenue even when we stop constraint generation for problem (3) at an intermediate iteration.

Problem (11) has a surprising connection to the earlier literature. Kunnumkal and Topaloglu (2010) propose problem (11) as a deterministic linear programming approximation to the network revenue management problem. The strongest result they can give for problem (11) is that the optimal objective value of this problem is greater than or equal to the optimal objective value of problem (3). Their result gives the impression that problem (3) is a stronger approximation for the network revenue management problem than problem (11) as it potentially yields a tighter upper bound, but our findings in this paper show that this is not the case and problems (3) and (11) are equivalent to each other. Another interesting observation is that Kunnumkal and Topaloglu (2010) obtain problem (11) by using Lagrangian relaxation to relax the capacity availability constraints in the dynamic programming formulation in (1). This observation naturally raises the question of whether using Lagrangian relaxation on the dynamic programming formulation of a general decision making problem, as is done in Kunnumkal and Topaloglu (2010), is equivalent to using affine approximations to the value functions, as is done in Adelman (2007). It is not too difficult to see that this is not true in general. To give a simple example, if the decisions made at one time period has no effect on the future decisions, then using a value function approximation of zero at each time period would still give the optimal policy since there is no need to use value functions to assess the impact of the current decisions on the future. So, we can obtain the optimal policy trivially by using affine value function approximations with zero slopes and zero intercepts. In contrast, relaxing certain constraints through Lagrangian relaxation at each time period may certainly result in suboptimal decisions, especially when there are integrality constraints on the decisions. Thus, using affine value function approximations is not equivalent to using Lagrangian relaxation in the dynamic programming formulation of a general decision making problem, but in the network revenue management setting, our results in this paper establish such an equivalence.

There may be other ways to show the equivalence between problems (3) and (11). Subsequent to our work, Vossen and Zhang (2012) gave an alternative proof for the equivalence between problems (3) and (11) by aggregating the decision variables in the dual of problem (3). In their work, Vossen and Zhang (2012) take problems (3) and (11) as given and show how to obtain the optimal solution to one problem by using the optimal solution to the other. In contrast, we follow a constructive approach, showing the equivalence between problems (3) and (11) as we construct problem (11).

7 Computational Experiments

In this section, we provide computational experiments that compare the solution times when we use constraint generation to solve problems (3) and (11).

7.1 Experimental Setup

In our test problems, we work with two different types of airline networks. In the first type of network, we have a single hub serving N spokes. There is a flight leg from the hub to each spoke and another one from each spoke to the hub. Thus, the number of flight legs is $2N$. It is possible to go from each origin to each destination in the airline network so that the number of possible origin-destination pairs is $N(N+1)$. Going from one spoke to another requires taking two flight legs through the hub. Going from a spoke to the hub or from the hub to a spoke is possible through a direct flight leg. Figure 2.a shows the structure of the first type of network with $N = 8$. In the second type of network, we have two hubs serving a total of N spokes. The first half of the spokes are connected to the first hub and the second half of the spokes are connected to the second hub. There is a flight leg from each spoke to a hub and another flight leg from the corresponding hub to each spoke. There are also two flight legs that connect the two hubs in two directions. Thus, the number of flight legs is $2N + 2$. It is possible to go from each origin to each destination in the airline network, in which case, the number of possible origin-destination pairs is $(N+1)(N+2)$. Going from one spoke to another requires taking two or three flight legs, whereas going from a spoke to a hub or from a hub to a spoke requires taking one or two flight legs. It is possible to go from one of the hubs to the other through a direct flight leg. Figure 2.b shows the structure of the second type of network with $N = 8$.

We have a high-fare and a low-fare itinerary associated with each origin-destination pair in the airline network. The fare associated with a high-fare itinerary is four times the fare associated with the corresponding low-fare itinerary. The arrival probabilities are calibrated so that the probability of getting a request for a high-fare itinerary increases over time, whereas the probability of getting a request for a low-fare itinerary decreases. Since the total expected demand for the capacity on flight leg i is given by $\sum_{t\in\mathcal{T}}\sum_{j\in\mathcal{J}} a_{ij}\, p_{jt}$, we measure the tightness of the leg capacities by

$$\alpha = \frac{\sum_{i\in\mathcal{L}}\sum_{t\in\mathcal{T}}\sum_{j\in\mathcal{J}} a_{ij}\, p_{jt}}{\sum_{i\in\mathcal{L}} c_i}.$$

In our test problems, we vary the number of time periods in the selling horizon over $\{600, 800, 1{,}000\}$, the number of spokes in the airline network over $\{8, 12\}$ and the tightness of the leg capacities over $\{1.0, 1.3, 1.6\}$. This experimental setup yields 18 test problems for each type of airline network.

7.2 Implementation of Constraint Generation

We experimented with a number of constraint generation strategies for solving problem (3). The following strategy consistently performed the best for our test problems. Letting $\{\tilde{\theta}_t : t \in \mathcal{T}\}$ and $\{\tilde{v}_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ be the solution to the current master problem, to check whether this solution violates any of constraints (3b) in problem (3), we solve problem (4) for each time period. If the optimal objective value of problem (4) exceeds $\tilde{\theta}_t - \tilde{\theta}_{t+1}$ for a time period t, then using $\tilde{x}_t$ and $\tilde{u}_t$ to denote an optimal solution to problem (4), constraint (3b) corresponding to $\tilde{x}_t \in \mathcal{X}$, $\tilde{u}_t \in \mathcal{U}(\tilde{x}_t)$ and $t \in \mathcal{T}$ is violated. We add this constraint to the master problem. Once we find five violated constraints, we stop looking for others. We solve the master problem with the added constraints and this concludes

one iteration of constraint generation. As the iterations progress, the constraints added in the earlier iterations may not be relevant any more and become loose. If the number of loose constraints exceeds 70% of the number of constraints in the master problem, then we remove all loose constraints from the master problem. Finally, Adelman (2007) shows that the decision variables $\{v_{it} : i \in \mathcal{L},\ t \in \mathcal{T}\}$ satisfy $v_{i1} \geq v_{i2} \geq \ldots \geq v_{i\tau}$ for all $i \in \mathcal{L}$ in an optimal solution to problem (3). We add all of these constraints to the master problem before we even start constraint generation. Adding these constraints at the beginning significantly speeds up the overall performance of constraint generation. The same observation is also made by Adelman (2007).

We solve problem (11) by using constraint generation as well. For our test problems, the best performing constraint generation strategy we found is as follows. We add all of constraints (11b), (11c) and (11e) to the master problem at the beginning. Therefore, we only look for violations of constraints (11d). After solving the master problem at the current iteration, we enumerate over all $i \in \mathcal{L}$, $j \in \mathcal{J}$ and $t \in \mathcal{T}$ to find which of constraints (11d) are violated by the solution to the current master problem. We add all of the violated constraints to the master problem and solve the master problem with the added constraints. This concludes one iteration of constraint generation. We note that although we solve problem (11) by using constraint generation, generating constraints for this problem is trivial.

7.3 Computational Results

Tables 1 and 2 summarize our main computational results, where the two tables respectively focus on the test problems with one and two hubs. In these tables, the first column shows the characteristics of the test problem by using the triplet (τ, N, α), where τ is the number of time periods in the selling horizon, N is the number of spokes and α characterizes the tightness of the leg capacities. Recalling that we have a high-fare and a low-fare itinerary for each origin-destination pair, if there are N spokes in the airline network, then the numbers of flight legs and itineraries are respectively $2N$ and $2N(N+1)$ for the airline networks with one hub, whereas the numbers of flight legs and itineraries are respectively $2N + 2$ and $2(N + 1)(N + 2)$ for the airline networks with two hubs. The second to fifth columns in Tables 1 and 2 show the performance of constraint generation when applied to problem (3). In particular, the second column shows the CPU seconds required to solve problem (3) to optimality. The third column shows the CPU seconds required to solve problem (3) with 1% optimality gap. The fourth column shows what percentage of the CPU seconds is spent on generating constraints. The remaining percentage of the CPU seconds is spent on solving the master problem. The fifth column shows the total number of constraints generated to solve problem (3) to optimality. The interpretations of the sixth, seventh, eighth and ninth columns are similar to those of the previous four, but these columns focus on constraint generation when applied to problem (11). The last two columns in Tables 1 and 2 compare the solution times for problems (3) and (11). In particular, the tenth column shows the ratio between the CPU seconds required to solve problems (3) and (11) to optimality. The eleventh column shows the ratio between the CPU seconds required to solve problems (3) and (11) with 1% optimality gap. Our computational implementation is carried out by using Gurobi 4.5 as the linear programming solver. In our computational experiments, we focus on demonstrating the computational savings obtained by solving problem (11) by using constraint generation instead of solving problem

(3). We do not test the performance of the policies obtained from these problems. Adelman (2007) and Topaloglu (2009) compare the performance of the policies with a variety of benchmark strategies and report quite favorable results.

The results in Tables 1 and 2 indicate that problem (11) provides significant computational savings over problem (3) in terms of CPU seconds. Over all of our test problems, the longest CPU seconds for problem (11) is 33 seconds, but problem (3) may take up to 1,931 seconds for some of the larger test problems with large number of time periods in the selling horizon and large number of spokes in the airline network. Similar observations apply if we are interested in obtaining a solution with 1% optimality gap. By using problem (11), we can obtain a solution with 1% optimality gap within 30 seconds in the worst case. For most of the test problems, we can obtain a solution with 1% optimality gap within three seconds. On the other hand, problem (3) may take up to 449 seconds to solve with 1% optimality gap. The ratio between the CPU seconds required to solve problems (3) and (11) can be as high as 135. The average of the ratios between the CPU seconds required to solve the two problems comes out to be 52. If we are interested in obtaining a solution with 1% optimality gap, then the average ratio between the CPU seconds is 71.

To get a feel for the problem characteristics that affect the CPU seconds for problems (3) and (11), we observe that the CPU seconds for both problems increase as the number of time periods in the selling horizon and the number of spokes increase, which is not surprising, since these problem characteristics directly affect the numbers of decision variables and constraints in problems (3) and (11). On the other hand, we observe that the CPU seconds for both problems also increase as the leg capacities get tighter. Although the numbers of decision variables and constraints do not depend on the tightness of the leg capacities, we need to generate more constraints to obtain the optimal solution and this translates into longer CPU seconds. It is also worthwhile to note that the largest values for the ratios between the CPU seconds required to solve problems (3) and (11) correspond to the test problems with tight leg capacities. Therefore, although it takes more time to solve either of the two problems as the leg capacities get tighter, problem (11) is affected less. It is encouraging that problem (11) provides the largest computational savings when problem (3) is particularly difficult to solve.

To sum up, our computational results indicate that using constraint generation to solve problem (11) is significantly more efficient than working with problem (3) directly. Problem (11) maintains its advantage across a variety of sizes of test problems and the gaps in the CPU seconds are most noticeable for the larger test problems with tighter leg capacities. Given these considerations, reducing problem (3) to problem (11) and solving the latter problem by using constraint generation is clearly a viable alternative to solving problem (3) by using constraint generation.

The CPU seconds reported for problem (11) in Tables 1 and 2 correspond to the case where we solve this problem by using constraint generation as described in Section 7.2. A natural question is how much benefit we obtain from using constraint generation to solve problem (11). To answer this question, Table 3 compares the CPU seconds for problem (11) when we solve this problem by using constraint generation and when we solve this problem directly by using a linear programming solver. The left and right sides of Table 3 respectively focus on the test problems with one and two hubs. The first column

in this table shows the characteristics of the test problem. The second and third columns respectively show the CPU seconds required to solve problem (11) to optimality and with 1% optimality gap by using constraint generation. These two columns are identical to the sixth and seventh columns in Tables 1 and 2. The fourth and fifth columns in Table 3 respectively show the CPU seconds required to solve problem (11) to optimality and with 1% optimality gap directly by using a linear programming solver. In other words, the CPU seconds in the fourth and fifth columns correspond to the case where we a priori construct all of the constraints in problem (11) and directly maximize the objective function of problem (11) subject to all of the constraints listed in this problem. The results in Table 3 show that it may be faster to solve problem (11) directly by using a linear programming solver when the number of spokes is small or when the leg capacities are not too tight. However, if the number of spokes in the airline network is large and the leg capacities are tight, then it is significantly faster to solve problem (11) by using constraint generation. The benefits from constraint generation are especially noticeable when the CPU seconds are on the large side.

Finally, Table 4 shows the CPU seconds for problem (11) on larger test problems. These test problems involve 20 or 24 spokes and a single hub. Similar to our other test problems, we vary the number of time periods in the selling horizon over $\{600, 800, 1{,}000\}$ and the tightness of the leg capacities over $\{1.0, 1.3, 1.6\}$. The first column in Table 4 shows the characteristics of the test problem. The second and third columns respectively show the CPU seconds required to solve problem (11) to optimality and with 1% optimality gap. The CPU seconds given in Table 4 correspond to the case where we solve problem (11) by using constraint generation and our results show that we can use constraint generation to obtain the optimal solution to problem (11) within several minutes. It turns out that even for the largest test problems, we can obtain the optimal solution to problem (11) within eight minutes. For the test problems with 24 spokes, when we try to solve problem (11) directly by using a linear programming solver, we would either not be able to obtain the optimal solution within 10 minutes or run out of two gigabytes of memory. Furthermore, we tried to solve problem (3) by using constraint generation, but we could not get a solution with 1% optimality gap within a time limit of two hours. Therefore, exploiting the equivalence between problems (3) and (11) and solving problem (11) by using constraint generation provides a viable approach for dealing with large test problems.

8 Conclusions

In this paper, we considered the approximate linear programming approach for network revenue management problems. This approach ends up with a linear program whose number of constraints increases exponentially with the number of flight legs. This linear program is commonly solved by using constraint generation. Each constraint can be generated by solving a separate integer program. The necessity to solve integer programs to generate constraints and the slow convergence behavior of constraint generation methods are practical drawbacks for using the approximate linear programming approach on network revenue management problems. Our goal in this paper was to address these drawbacks. We showed that we can generate constraints for the linear program by solving minimum-cost network flow problems. Furthermore, by exploiting the minimum-cost network flow structure, we showed that we can a priori reduce the number of constraints in the linear program from exponential in

the number of flight legs to linear. Computational experiments indicated that our results can provide substantial savings in terms of computation time.

The approximate linear programming approach finds applications in other settings, such as inventory distribution, vehicle routing and joint replenishment. In these settings, it is customary to formulate the problem as a dynamic program with a high-dimensional state variable and use affine approximations to the value functions. It would be of interest to explore whether the results that we showed in this paper can be extended to enhance the computational performance of the approximate linear programming approach when applied in settings other than network revenue management.

Acknowledgements

The authors thank two anonymous referees, the associate editor and the area editor for their helpful comments. This work was supported in part by National Science Foundation Grants CMMI-0825004 and CMII-0969113.

References

Adelman, D. (2007), 'Dynamic bid-prices in revenue management', *Operations Research* **55**(4), 647–661.

Farias, V. F. and van Roy, B. (2007), An approximate dynamic programming approach to network revenue management, Technical report, Stanford University, Department of Electrical Engineering.

Kunnumkal, S. and Topaloglu, H. (2010), 'Computing time-dependent bid prices in network revenue management problems', *Transportation Science* **44**(1), 38–62.

Meissner, J. and Strauss, A. K. (2008), Network revenue management with inventory-sensitive bid prices and customer choice, Technical report, Lancaster University Management School, Department of Management Science.

Puterman, M. L. (1994), *Markov Decision Processes*, John Wiley and Sons, Inc., New York.

Ruszczynski, A. (2006), *Nonlinear Optimization*, Princeton University Press, Princeton, New Jersey.

Simpson, R. W. (1989), Using network flow techniques to find shadow prices for market and seat inventory control, Technical report, Massachusetts Institute of Technology Flight Transportation Laboratory Memorandum M89-1, Cambridge, MA.

Talluri, K. and van Ryzin, G. (1998), 'An analysis of bid-price controls for network revenue management', *Management Science* **44**(11), 1577–1593.

Talluri, K. and van Ryzin, G. (1999), 'A randomized linear programming method for computing network bid prices', *Transportation Science* **33**(2), 207–216.

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

Vanderbei, R. (1997), *Linear Programming: Foundations and Extensions*, Kluwer's International Series.

Vossen, T. W. M. and Zhang, D. (2012), A dynamic disaggregation approach to approximate linear programs for network revenue management, Technical report, University of Colorado at Boulder, Boulder, CO.

Williamson, E. L. (1992), Airline Network Seat Control, PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA.

Zhang, D. and Adelman, D. (2009), 'An approximate dynamic programming approach to network revenue management with customer choice', *Transportation Science* **42**(3), 381–394.

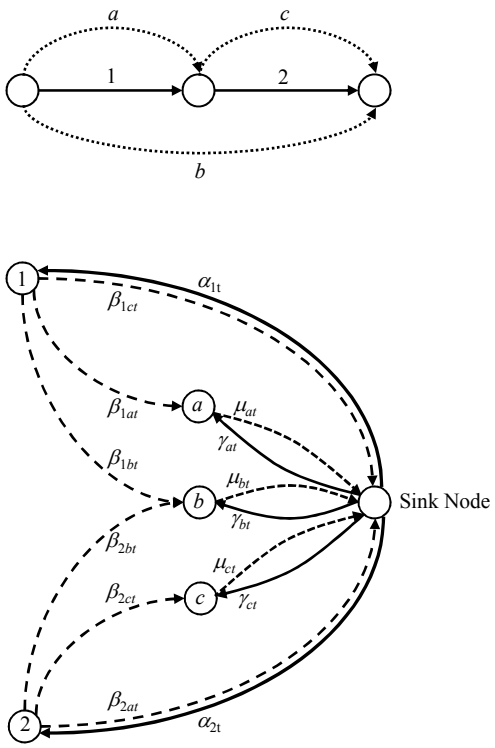

Figure 1: Minimum-cost network flow problem corresponding to problem (6).

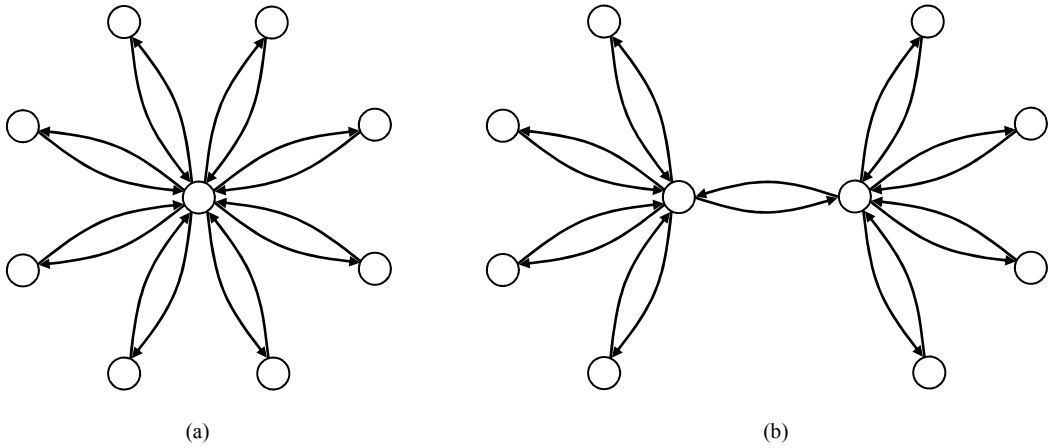

Figure 2: Structure of the airline network with one and two hubs for the case with $N = 8$.

Test problem (τ, N, α)	Cons. gen. for prob. (3) Total secs.	1% secs.	% cons. gen.	# cons.	Cons. gen. for prob. (11) Total secs.	1% secs.	% cons. gen.	# cons.	Ratio of secs. Total secs.	1% secs.
(600, 8, 1.0)	27.95	12.70	14.45	1,587	1.96	0.48	16.84	2,181	14.26	26.46
(600, 8, 1.3)	134.61	37.84	8.86	3,754	3.09	0.53	11.33	2,190	43.56	71.40
(600, 8, 1.6)	163.79	61.85	7.68	4,640	2.48	2.11	12.90	2,241	66.04	29.31
(600, 12, 1.0)	181.00	23.34	14.41	3,616	9.09	1.22	8.03	6,978	19.91	19.13
(600, 12, 1.3)	476.16	114.73	11.51	7,007	14.83	13.06	5.12	8,107	32.11	8.78
(600, 12, 1.6)	582.61	159.20	11.37	7,894	15.64	14.02	4.86	7,480	37.25	11.36
(800, 8, 1.0)	33.06	22.83	10.74	1,793	2.49	0.75	17.27	2,066	13.28	30.44
(800, 8, 1.3)	232.19	72.45	6.88	4,719	2.96	0.83	15.88	2,176	78.44	87.29
(800, 8, 1.6)	298.00	117.37	5.47	6,137	3.21	0.81	13.40	2,241	92.83	144.90
(800, 12, 1.0)	266.51	42.30	12.28	4,315	9.47	1.81	10.56	7,546	28.14	23.37
(800, 12, 1.3)	773.73	238.70	8.96	7,403	12.69	1.97	7.96	7,309	60.97	121.17
(800, 12, 1.6)	975.30	306.42	8.89	9,267	22.15	20.74	4.61	7,471	44.03	14.77
(1000, 8, 1.0)	58.15	38.53	9.36	2,231	2.72	1.00	20.59	2,336	21.38	38.53
(1000, 8, 1.3)	376.62	118.89	4.54	5,593	3.79	1.02	14.78	2,173	99.37	116.56
(1000, 8, 1.6)	451.43	201.08	4.39	7,166	3.34	1.01	16.47	2,268	135.16	199.09
(1000, 12, 1.0)	378.42	63.48	9.46	4,459	11.39	2.39	10.80	7,750	33.22	26.56
(1000, 12, 1.3)	1243.83	414.68	6.35	9,356	22.09	3.21	5.57	7,779	56.31	129.18
(1000, 12, 1.6)	1556.90	408.71	6.23	11,265	20.84	2.60	5.95	7,476	74.71	157.20
Average									52.83	69.75

Table 1: Computational results for the test problems with one hub.

Test problem (τ, N, α)	Cons. gen. for prob. (3) Total secs.	1% secs.	% cons. gen.	# cons.	Cons. gen. for prob. (11) Total secs.	1% secs.	% cons. gen.	# cons.	Ratio of secs. Total secs.	1% secs.
(600, 8, 1.0)	71.60	14.50	13.63	2,530	3.43	0.71	11.95	3,963	20.87	20.42
(600, 8, 1.3)	188.40	46.57	7.99	5,162	3.49	0.69	12.03	3,253	53.98	67.49
(600, 8, 1.6)	249.05	64.55	7.69	6,447	5.45	0.70	7.71	3,212	45.70	92.21
(600, 12, 1.0)	205.05	27.80	13.76	3,820	5.57	1.56	14.90	10,794	36.81	17.82
(600, 12, 1.3)	645.48	184.92	10.31	7,636	16.54	1.56	5.02	10,108	39.03	118.54
(600, 12, 1.6)	893.31	208.37	13.87	9,747	30.57	21.17	2.72	9,654	29.22	9.84
(800, 8, 1.0)	115.27	25.83	12.01	3,049	3.89	1.03	13.88	3,905	29.63	25.08
(800, 8, 1.3)	280.69	81.41	6.91	5,420	5.37	0.98	10.06	3,234	52.27	83.07
(800, 8, 1.6)	470.34	121.14	5.19	8,502	5.32	1.00	10.34	3,234	88.41	121.14
(800, 12, 1.0)	269.02	49.32	9.07	3,464	8.41	2.16	13.08	10,896	31.99	22.83
(800, 12, 1.3)	1108.39	239.38	8.43	9,322	29.66	2.18	3.71	9,961	37.37	109.81
(800, 12, 1.6)	1587.84	396.69	9.89	11,686	33.38	30.93	4.37	9,639	47.57	12.83
(1000, 8, 1.0)	119.30	43.80	14.74	2,583	3.55	1.32	19.44	3,811	33.61	33.18
(1000, 8, 1.3)	422.91	130.35	5.51	6,146	5.79	1.30	12.61	3,183	73.04	100.27
(1000, 8, 1.6)	758.49	196.56	3.92	10,524	7.58	1.29	8.84	3,233	100.06	152.37
(1000, 12, 1.0)	405.09	76.06	6.81	4,148	7.80	2.82	17.82	10,946	51.93	26.97
(1000, 12, 1.3)	1909.78	411.63	5.73	12,112	22.58	2.83	6.16	9,235	84.58	145.45
(1000, 12, 1.6)	1931.23	449.95	9.80	15,925	30.44	2.87	4.53	9,697	63.44	156.78
Average									51.08	73.12

Table 2: Computational results for the test problems with two hubs.

Test problem (τ, N, α)	Cons. gen. for prob. (11) Total secs.	Cons. gen. for prob. (11) 1% secs.	Dir. solut. for prob. (11) Total secs.	Dir. solut. for prob. (11) 1% secs.
(600, 8, 1.0)	1.96	0.48	1.01	0.81
(600, 8, 1.3)	3.09	0.53	2.94	1.82
(600, 8, 1.6)	2.48	2.11	3.56	2.09
(600, 12, 1.0)	9.09	1.22	8.52	3.00
(600, 12, 1.3)	14.83	13.06	22.39	7.62
(600, 12, 1.6)	15.64	14.02	36.01	9.18
(800, 8, 1.0)	2.49	0.75	1.79	1.46
(800, 8, 1.3)	2.96	0.83	2.82	1.95
(800, 8, 1.6)	3.21	0.81	5.99	3.61
(800, 12, 1.0)	9.47	1.81	8.67	3.33
(800, 12, 1.3)	12.69	1.97	18.21	5.72
(800, 12, 1.6)	22.15	20.74	58.15	15.43
(1000, 8, 1.0)	2.72	1.00	4.35	3.55
(1000, 8, 1.3)	3.79	1.02	5.32	3.21
(1000, 8, 1.6)	3.34	1.01	8.20	3.87
(1000, 12, 1.0)	11.39	2.39	9.17	4.82
(1000, 12, 1.3)	22.09	3.21	45.38	10.62
(1000, 12, 1.6)	20.84	2.60	82.29	11.27

Test problem (τ, N, α)	Cons. gen. for prob. (11) Total secs.	Cons. gen. for prob. (11) 1% secs.	Dir. solut. for prob. (11) Total secs.	Dir. solut. for prob. (11) 1% secs.
(600, 8, 1.0)	3.43	0.71	1.79	0.78
(600, 8, 1.3)	3.49	0.69	3.48	1.62
(600, 8, 1.6)	5.45	0.70	6.06	2.42
(600, 12, 1.0)	5.57	1.56	5.51	1.90
(600, 12, 1.3)	16.54	1.56	27.16	9.43
(600, 12, 1.6)	30.57	21.17	47.57	12.61
(800, 8, 1.0)	3.89	1.03	2.16	1.87
(800, 8, 1.3)	5.37	0.98	5.71	1.81
(800, 8, 1.6)	5.32	1.00	8.11	4.17
(800, 12, 1.0)	8.41	2.16	6.18	3.42
(800, 12, 1.3)	29.66	2.18	56.74	6.60
(800, 12, 1.6)	33.38	30.93	83.88	17.36
(1000, 8, 1.0)	3.55	1.32	4.96	1.47
(1000, 8, 1.3)	5.79	1.30	4.51	2.35
(1000, 8, 1.6)	7.58	1.29	10.52	4.36
(1000, 12, 1.0)	7.8	2.82	8.71	3.43
(1000, 12, 1.3)	22.58	2.83	53.38	7.34
(1000, 12, 1.6)	30.44	2.87	110.58	14.19

Table 3: CPU seconds for problem (11) when we solve this problem by using constraint generation and when we solve this problem directly by using a linear programming solver.

Test problem (τ, N, α)	Cons. gen. for prob. (11) Total secs.	Cons. gen. for prob. (11) 1% secs.
(600, 20, 1.0)	34.77	3.69
(600, 20, 1.3)	42.07	9.19
(600, 20, 1.6)	54.17	12.93
(600, 24, 1.0)	151.56	5.02
(600, 24, 1.3)	124.70	20.69
(600, 24, 1.6)	120.11	50.57

Test problem (τ, N, α)	Cons. gen. for prob. (11) Total secs.	Cons. gen. for prob. (11) 1% secs.
(800, 20, 1.0)	37.69	4.71
(800, 20, 1.3)	62.11	5.06
(800, 20, 1.6)	74.30	11.21
(800, 24, 1.0)	80.79	6.84
(800, 24, 1.3)	96.69	18.66
(800, 24, 1.6)	454.20	26.17

Test problem (τ, N, α)	Cons. gen. for prob. (11) Total secs.	Cons. gen. for prob. (11) 1% secs.
(1000, 20, 1.0)	46.67	5.96
(1000, 20, 1.3)	78.63	6.00
(1000, 20, 1.6)	66.77	12.13
(1000, 24, 1.0)	85.91	8.69
(1000, 24, 1.3)	164.79	8.93
(1000, 24, 1.6)	143.18	26.87

Table 4: CPU seconds for problem (11) for large test problems.