NETWORK FLOWS IN HOTEL YIELD MANAGEMENT

by

Dietrich Chen
Network Flows in Hotel Yield Management

Dietrich Chen

Abstract

Taking a common linear programming formulation for maximizing revenue in the hotel industry based on demand forecasts, we show that integrality comes for free by giving an equivalent network flow formulation, and use the latter to solve for optimal room allocation in a typical hotel. Computational requirements are such that these decisions could be made in real-time. Computational results are presented as well as natural extensions such as upgrading and group reservations.

linear programming * network flows * hotel industry * bid-price * group reservations * upgrading

Williamson [7] formulates the problem of maximizing revenue in the airline industry as a linear program and observes the integrality property of the optimal solution. We apply the linear programming approach to the hotel industry and prove that the optimal solution will always be integral by providing an equivalent network flow formulation. In addition, we can include extensions to the model such as upgrading (by adding side constraints) and group reservations. Due to the nature of the network flow formulation, computations are very efficient. Computational results for a typical hotel are presented.

We will proceed to state the mathematical model and extensions in the first two sections. An equivalent network flow formulation of the linear programming model is given in section 3 followed by a discussion of our computational experiments (section 4). Finally, we suggest some interesting research questions in section 5.

*School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY 14853-3801
(didi@orie.cornell.edu)
1 Mathematical Model

For a survey of different yield management techniques we refer the interested reader to Weatherford and Bodily [6] and Kimes [4]. For the remainder of the paper, we assume that room rates are set and demand is deterministic, i.e. we have perfect information regarding customer demand classified by day of arrival, length of stay, and rate class. We are thus solving a subproblem within an overall approach that integrates forecasting and reservation systems.

The problem now is easy to state: find an optimal mix of room allocations so as to maximize revenue from selling rooms in a hotel. We neglect additional revenue such as food and beverage, meeting and conference space, etc.

The model is expressed in terms of the following variables:

\[ i, l, j = \text{date indices} (i, l, j = 1, \ldots, N), \]
\[ i \text{ and } l \text{ usually refer to the arrival day whereas } j \text{ denotes the departure day;} \]
\[ k = \text{rate class index} (k = 1, \ldots, K); \]
\[ x_{ijk} = \text{number of rooms sold to guests arriving on day } i \text{ and departing on day } j, i < j, \text{at rate } k; \]
\[ c_k = \text{rate class } k; \]
\[ b_i = \text{authorized capacity for the hotel on day } i; \]
\[ d_{ijk} = \text{demand for the number of rooms sold to guests arriving on day } i \text{ and departing on day } j \text{ at rate } k \]
\[ \tau = \text{maximum length of stay.} \]

Hence, we want to choose \( x_{ijk} \) so as to
\[
\text{maximize } \sum_{i, j, k} (j - i) c_k x_{ijk} \\
\text{subject to:}
\]
\[
\forall i : \sum_{t \leq i} \sum_{i < j} \sum_k x_{ijk} \leq b_i, \\
0 \leq x_{ijk} \leq d_{ijk}.
\]
\[x_{ijk} \text{ integer.}\]  

Clearly, if a customer requests a room with an associated room rate \(c_k\) from day \(i\) to day \(j\), the contribution to revenue is \((j - i)c_k x_{ijk}\), hence (1).

Each day the total number of guests in the hotel cannot exceed the authorized capacity level for that day. If no overbooking is permitted, \(b_i\) simply represents hotel capacity, i.e. total number of rooms in the hotel. Note that by adjusting \(b_i\) we could allow for overbooking. Overbooking sometimes leads to overselling which occurs when more guests with valid reservations show up than we have rooms in the hotel. In this case, guests have to be walked, i.e. they are denied a room in the hotel and referred to another hotel (often the cost of walking is covered by the hotel that denied the reservation, even though walking policies vary from country to country).

Constraints (2) are the capacity constraints. The total number of guests that arrive on day \(i\) plus the total number of guests that arrived prior to day \(i\) and stay at least until day \(i + 1\) must be less than or equal to the authorized capacity level \(b_i\) for that day.

Constraints (3) are saying merely that we cannot sell more rooms than there is demand. And finally, we cannot sell a fraction of a room (constraints (4)).

We will refer to this integer program as (IP). It is not clear \emph{a priori} that if we relax (IP) to a linear program (LP) by leaving out the integrality constraints (4), that the solution will always be integral. However, in section 3 we show that constraints (4) are indeed superfluous. Therefore, (IP) can be solved in polynomial time if only for the fact that the ellipsoid method (Khachiyan [3]) is an algorithm for linear programming that runs in polynomial time.
2 Extensions

The following extensions to our model are easily incorporated and allow us to examine a more realistic setting.

2.1 Bid-Prices

Solving the (LP) provides us with more information than just the optimal mix of rooms. In particular, we can compute the shadow prices for the capacity constraints in (2).

These shadow prices tell us how much an additional room is worth on a particular day. Williamson [7] calls these shadow prices “bid-prices.” Suppose we were able to expand the hotel’s capacity by one additional room. If we auction this room off to potential customers, we should start off with the bid-price to ensure that the price we receive for that room will be profitable for the hotel. Alternatively, one can think of the bid-price as the floor of a potential discount that we would be willing to quote to a customer. A rate below this threshold value would not be profitable to the hotel. For instance, if a hotel has four different room rates of, say, $200, $160, $130, and $90, and the bid-price for a particular day is $100, then all rates on this day should be available except the $90 room rate which is below the bid-price of $100.

When a customer asks for a room for consecutive days in the future, the average bid-price for the capacity constraints for these dates specifies the minimum profitable rate. So suppose a customer wants to stay for three nights. We compute the bid-prices for the particular days as $100, $170, and $150. To remain profitable, the contribution to revenue from that customer should be at least $420. In order to quote the customer a uniform room rate for all three nights, we compute the average bid-price, $140. Hence, for this particular customer, rates $160 and $200 remain open while we close down rates $130 and $90.

2.2 Upgrading

In the airline industry, Belobaba [1] extends his EMSR (Expected Marginal Seat Revenue) model to include the possibility of a vertical shift on the part of a passenger that is denied a seat on a particular flight. When a passenger is denied a seat for the requested fare class, he may choose to book a higher fare class on the same flight (if available): that is called vertical shift. Consequently, the passenger will have to pay the more
expensive fare to get on the flight. Belobaba shows how these upgrade probabilities affect the protection levels for higher fare classes as determined by his EMSR model.

Within the linear programming framework, we assume that demand for certain rates is distinct, i.e. a customer that is denied a request represents lost revenue. However, it is still possible to incorporate a slightly different notion of upgrading into this methodology.

Suppose we have distinct types of rooms in a hotel (e.g. standard rooms, deluxe rooms, and suites). Furthermore, assume that there is a complete preference relation that reflects the price structure of the rooms. More precisely, any customer would prefer a room of type 1 to a room of type 2 and so forth, where room type 1 is more expensive than room type 2 and so on. In addition, assume that the hotel only has \( cap_k \) number of rooms for room type 1.

Clearly, if we experience low demand for room type 1 say, and more demand for type 2 than we have rooms of type 2 we would like to upgrade customers to type 1 so as not to turn away business. We assume that no customer wants to get downgraded. In general, we may upgrade a customer who requests a type \( k \) room to rooms of type 1,\ldots,k – 1.

Therefore, upgrading as we define it reflects the notion of a complimentary upgrade on the part of the hotel: if a customer shows up with a reservation for room type \( k \) and all rooms of type \( k \) are occupied, we upgrade him to a better room if our demand forecast indicates that not all rooms of type 1,\ldots,k will be utilized.

Thus, to incorporate upgrading into our model, we want to add the following constraints:

\[
\forall i \quad \forall k : \quad \sum_{l<i} \sum_{i<j} x_{ijk} \leq \sum_{m \leq k} cap_m. \tag{5}
\]

Note that we do not have to add this constraint for the last desirable room type \( k \).

In addition, an overbooking level is only specified for the total number of rooms in a hotel and not for each room type. By doing this we can ensure that only the least profitable customers have to be walked in case we oversell the hotel.

To take this one step further, suppose that in addition to different room types we also have distinct rates
within each type (different rates for the same room can be made available for e.g. senior citizens, corporate customers, members of AAA, preferred customers, etc.).

Hence, assume you have \( r(k) \) different rates for room type \( k \). Constraints of type (5) then become:

\[
\forall i \quad \forall k : \quad \sum_{l \leq i} \sum_{i < j \leq r(k)} \sum_{p \leq r(k)} \sum_{m \leq k} x_{ljkp} \leq \sum_{m \leq k} cap_m,
\]

where \( x_{ljkp} \) of course, denotes the number of rooms of type \( k \) sold to customers who arrive on day \( l \) and stay until day \( j \) at a rate of class \( p \). The objective function (1) as well as constraints (2) and (3) are adjusted accordingly.

Note that we need only a preference relation with respect to room types. It is possible to have rates in type \( k \) that are priced higher than some rooms in type \( l \) even though rooms of type \( l \) are more desirable (according to our preference relation) than rooms of type \( k \).

2.3 Groups

Svrcek [5] extends the (IP) formulation to include group reservations when the following information about the group is available:

\[
i^* \quad = \quad \text{day of arrival};
\]

\[
\lambda_g \quad = \quad \text{length of stay};
\]

\[
\mu_g \quad = \quad \text{group size};
\]

\[
c_g \quad = \quad \text{group rate};
\]

\[
x_g \quad = \quad 0\text{-}1 \text{ variable indicating acceptance of the group.}
\]

The formulation now becomes:

\[
\text{maximize } \sum_{i,j,k} (j - i)c_kx_{ijk} + \lambda_g c_g \mu_g x_g
\]
subject to:

\[ \forall i \notin \{i^*, \ldots, i^* + \lambda_y \} : \sum_{i \leq i} \sum_{i < j} \sum_{k} x_{ijk} \leq b_i, \]

\[ \forall i \in \{i^*, \ldots, i^* + \lambda_y \} : \sum_{i \leq i} \sum_{i < j} \sum_{k} x_{ijk} + \mu_y x_y \leq b_i, \]

\[ 0 \leq x_{ijk} \leq d_{ijk}, \]

\[ x_{ijk} \text{ integer}, \]

\[ x_y \in \{0, 1\}. \]

In practice, however, the group rate is usually not predetermined but rather negotiated. During rate negotiations it would be helpful to know what the minimum profitable room rate for a given group is.

A simple method to compute this threshold value is to calculate the revenue associated with the optimal room allocation when the group is not taken into account and deduct from this value the revenue when we do take the group into account by adjusting the authorized capacities for the days in question. The net value (if positive) is then used to compute the minimum profitable room rate for the group by dividing it by the number of days the group wants to stay and the number of rooms it requests for these days.

One can also try to estimate additional revenue that the group is likely to incur, such as food and beverage, renting conference rooms and banquets, etc. If such an estimate can be made, it can be factored into the computation of the minimum profitable room rate.

3 Network Flow Formulation

The previous sections developed a model for yield management in the hotel industry. The sizes of the linear programming versions, however, are still too large if we want to solve them efficiently in practice. For instance, if we allow customers to make reservations up to 6 months in advance, with a maximum length of stay of 2 weeks, in a hotel with 15 different rates, we are dealing with about 40,000 decision variables. At resorts, where customers usually stay longer and sometimes book up to a year in advance, the number of decision variables can blow up to more than 100,000.
By formulating an equivalent network flow formulation, we prove the integrality property of the optimal solution of the (LP). Moreover, it allows us to solve for the optimal room mix in a typical hotel in real-time.

In the airline industry, Glover et al. [2] proposed a network flow model with special side constraints to maximize revenue by determining optimal passenger itineraries and fare class-mix on each segment of the flight.

Based on their work, we develop a network flow formulation for determining an optimal room allocation for a hotel with multiple-night stays and multiple rates.

Let $G = (V, A)$ be a directed graph with multiple arcs. Multiple forwards arcs $(i, j)_k$ connect nodes $i$ and $j$ whenever $j - i \leq \tau$ and $i < j$. The number of multiple forwards arcs between any two nodes is determined by the number of different rate classes. We only have single backward arcs of the type $(i + 1, i)_0$.

We define a profit function $\phi$ on the set of arcs as follows:

$$\forall (i, j)_k \in A : \quad \phi((i, j)_k) = \begin{cases} (j - i)c_k & j - i \leq \tau \\ 0 & i > j. \end{cases}$$

Figure 1 displays a small example of $G = (V, A)$.

![Figure 1: Example of $G = (V, A)$](image)

We will now formulate the network flow problem on $G = (V, A)$ under the profit function $\phi$.

Nodes in the network represent days. Flow on forward arcs $(i, j)_k$ between nodes $i$ and $j$ ($i < j$) represents the amount of rooms we sell to customers that arrive on day $i$ and stay until day $j$ with an associated room rate $k$.

Flow $x_{ijk}$ on forward arcs is capacitated by $d_{ijk}$. Flow $x_{i+1,i}$ on backward arcs is capacitated by $b_i$. 

8
The goal is to find a feasible flow (flow conservation at each node in the network holds) that maximizes revenue.

It is easy to see that this max revenue (min cost) network flow formulation is equivalent to our original model presented in section 1.

**Proposition 1** Finding a feasible flow on $G = (V, A)$ that maximizes revenue under the profit function $\phi$ is equivalent to solving (LP).

**Proof.** Certainly, by construction the objective function is the same since no revenue is associated with backward arcs.

We can show inductively how the flow conservation constraints on each node imply constraints of type (2). More precisely, given that we can generate the first $i - 1$ constraints in (2) from the first $i - 1$ flow conservation constraints, one simply deducts the $(i - 1)$-th constraint in (2) from the $i$-th flow conservation constraint to produce the $i$-th constraint in (2). For day 1, constraint (2) is trivially satisfied.

![Flow Conservation on Node i](image)

The $i - 1$-th constraint in (2) is:

$$\sum_{l \leq i - 1} \sum_{i - 1 < j} \sum_k x_{ljk} = x_{i,i-1} \leq b_{i-1},$$

whereas flow conservation on node $i$ specifies that (see Figure 2):

$$\sum_{l<i} \sum_k x_{lki} + x_{i+1,i} - \sum_{i<j} \sum_k x_{ijk} = 0.$$
Now, subtracting (7) from (6) yields:

\[
\sum_{l \leq i-1} \sum_{i < j} \sum_{k} x_{ijk} - \underbrace{\sum_{l < i} \sum_{k} x_{ljk} - x_{i+1,i}}_{\Sigma_{l \leq i-1} \Sigma_{i < j} \sum_{k} x_{ijk}} + \sum_{i < j} \sum_{k} x_{ijk} = 0
\]

\[
\Rightarrow \sum_{l \leq i} \sum_{i < j} \sum_{k} x_{ijk} = x_{i+1,i} \leq b_i,
\]

which is exactly the \(i\)-th constraint in (2).

Similarly one can generate the flow conservation constraints in our network model from constraints of type (2). The correspondance for constraints (3) and (4) to the network formulation is trivial. Thus, equivalence of both models has been established.

Note that without loss of generality, we can assume the parameters \(c_k, b_i,\) and \(d_{ijk}\) to be integral.

**Proposition 2** An optimal solution to \((LP)\) is always integral.

**Proof.** Immediate with Proposition 1.

To compute the bid-prices using the network flow formulation, we observe that the shadow prices for constraints (2) correspond to the reduced costs of the backward arcs in our network model which can be simply calculated by subtracting the dual variable \(w_i\) for node \(i\) from the dual variable \(w_{i+1}\) for node \(i + 1\).

## 4 Computational Results

We have tested the network flow formulation on randomly generated demands for a hotel with 250 rooms. Computations were done on a Sun Sparcstation 10 using CPLEX as a solver. CPU-times in sec. are reported which should be comparable to those obtained by the newest generation of PCs. Reading the input took less than 3 CPU-time seconds while writing the solution to an output file was accomplished usually within 1/10th of a CPU-time second.

Table 1 compares CPU-times obtained from solving the network flow formulation to the linear programming formulation. Instances marked with an * indicate that the formulation was too large which caused our
system to run out of memory. Most of the linear programs could not be solved when we specified realistic
input. Conversely, all network flow formulations were solved within seconds.

| Number of Days | 360 270 270 180 | 360 270 270 180 |
| Max. Length of Stay | 21 21 28 28 | 21 21 28 28 |
| Number of Rates | 15 15 15 15 | 10 10 10 10 |
| LP CPU-time | * * * * | * 23.9 * 25.3 |
| Network Flow CPU-time | 11.5 7.5 12.4 6.4 | 6.8 4.9 6.4 4.1 |

Table 1: Network Flow vs. LP Computations (CPU-times in sec.)

In practice, when a reservation is accepted, the demand forecast is updated accordingly. Since we update
reservation by reservation, the new demand forecast will be quite similar. Thus, reoptimization should be
almost even more efficient when using the current optimal solution as a starting point.

5 Future Directions

As a two-phase approach the validity of our model hinges on good demand forecasting. Furthermore, as a
ture real-time approach, forecasting necessarily needs to be fast and efficient.

We may make use of our efficient model formulation to investigate, e.g. what effect forecasting error has
on the optimal solution (when we have perfect information); what effect does a perturbation on the data
(input and/or parameters) have on the optimum. In the same spirit, a post-optimality analysis may give us
more information on what kind of data we should collect and be able to forecast.

In addition, there has been very little published on group reservations. Both forecasting methods and
inventory management methodologies need to be addressed. Extensions to existing group models should be
investigated, e.g. how to account for early and late departures.

Acknowledgments

The author wishes to thank Peter L. Jackson and Sheryl E. Kimes for many helpful discussions.
References


