Restaurant Revenue Management

Dimitris Bertsimas* Romy Shioda †

December 2000, revised August, 2001

Abstract

We develop two classes of optimization models in order to maximize revenue in a restaurant, while controlling average waiting time as well as perceived fairness, that may violate the first-come-first-serve (FCFS) rule. In the first class of models, we use integer programming, stochastic programming and approximate dynamic programming methods to decide dynamically when, if at all, to seat an incoming party during the day of operation of a restaurant that does not accept reservations. In a computational study with simulated data, we show that optimization based methods enhance revenue relative to the industry practice of FCFS by 0.11% to 2.22% for low load factors, by 0.16% to 2.96% for medium load factors, and by 7.65% to 13.13% for high load factors, without increasing and occasionally decreasing waiting times compared to FCFS. The second class of models addresses reservations. We propose a two step procedure: use a stochastic gradient algorithm to decide a priori how many reservations to accept for a future time and then use approximate dynamic programming methods to decide dynamically when, if at all, to seat an incoming party during the day of operation. In a computational study involving real data from an Atlanta restaurant, the reservation model improves revenue relative to FCFS by 3.5% for low load factors and 7.3% for high load factors.

*Boeing Professor of Operations Research, Sloan School of Management, Rm. E53-363, Massachusetts Institute of Technology, Cambridge, Mass. 02139. dbertsim@mit.edu.

†Operations Research Center, Massachusetts Institute of Technology, Cambridge, Mass. 02139. romy@mit.edu.

1 Introduction

Maximizing efficiency is of utmost importance for large and popular restaurants to increase their profits and to remain competitive. This can explain the surge in the usage of POS (point of sales) softwares that track the arrival time, size and order of each customer. Although floor managers can utilize these tools as an aid to better estimate the remaining service time of the customers and to see which tables need to speed up their service, their seating policy is mainly based on intuition brought on by experience. In most cases, they follow a simple first-come-first-serve (FCFS) policy.

The challenge of a floor manager is to decide when and where to seat each arriving customer. If there are only tables of four available and a party of two enters, does he seat the party at the larger table or reserve if for a larger more revenue producing party? In addition, if the restaurant takes reservations, he needs to further decide how to seat walk-in customers so that they would not take away tables from the reservation customers while considering the possibility of no-shows. These are important practical issues for restaurant managers, where in some cases a good floor manager can make the difference of several hundred dollars per night [7]. Thus, a tool that can help floor managers better make these decisions would be of significant value to a restaurant.

With all the data that is collected by the POS software, a revenue maximizing seating policy can be utilized. We believe the key lies in nesting - whether parties are seated at tables that can seat larger parties. The present paper stems from the belief that restaurants can increase their revenue by optimizing their nesting decisions, i.e., when to save tables in anticipation for larger parties, even when there are smaller parties currently in queue. We develop two classes of optimization models in order to maximize revenue in a restaurant, while controlling average waiting time as well as perceived fairness, that is the degree of violation of the first-come-first-serve (FCFS) rule. In the first class of models, we use integer programming, stochastic programming and approximate dynamic programming methods to decide dynamically when, if at all, to seat an incoming party *during the day of operation* of a restaurant that does not accept reservations. In the second class of models, we use a stochastic gradient algorithm to decide when, if at all, to accept a reservation for *a future time* and also incorporate reservations in a dynamic model. We illustrate using both simulated and real data, that our models lead to significant revenue enhancements relative to FCFS.

Literature

Revenue management in general is the practice of maximizing a company's revenue by optimally choosing which customers to serve. It has been used extensively in the airline, hotel, and car rental industry.

McGill and van Ryzin [10] give a comprehensive overview of the history of revenue management in transportation, where it has had the greatest impact.

In the case of restaurants, restaurant managers want to allocate their tables by seating the largest possible party at each table - assuming the total bill increases with party size. However, they need to also consider seating small parties at large tables when the larger parties are not expected to arrive in the near future, because they would rather seat them than having an empty table. Thus, the challenge is to understand the demand flow throughout the day of each type of customer and optimize the allocation of the tables among them.

Unlike the widespread application of revenue management methods in airlines, hotels, and rental cars, the number and depth of studies on revenue management in restaurants have been comparatively slim. Kimes et. al. [4], [5], [6] have been one of the first to directly address the issue of restaurant revenue management. They built a strategic framework for applying revenue management for restaurants to increase demand, and thus revenue, by effective duration management and demand-based pricing. They proposed using the revenue per available seat hour (RevPASH := revenue accrued in a given time interval divided by the number of seats available during that time) as the restaurant's performance metric. This value is calculated for each time period of each day, in order to identify times that the RevPASH is low. Similarly, Sill et. al. [13], [14] proposes the use of Capacity-management Science (CMS)® as a systematic method of assessing the restaurant's capacity potential and process efficiency. CMS® involves monitoring every component of the service and production delivery process with quantifiable measurements to improve customer satisfaction, enhance employee work life and increase profit.

Although not specified under the name of revenue management, many other approaches have been proposed to increase the revenue of restaurants. Vakharia et. al. [15] developed models and heuristics to find the best trade-off between wages and hour preferences to minimize the cost of employees while maintaining employee satisfaction. Quain et. al. [12] and Muller [11] addressed managerial factors that may improve the efficiency of restaurants, such as realizing profit centers, dispersing demand, decreasing operating hours and decreasing service time by making the restaurant operational procedures as efficient as possible.

Most of the studies in this area address issues concerning the overall management of the restaurant. However, to the best of our knowledge, there are no studies on a mathematically rigorous dynamic seating model.

Our objective in the present paper is to develop and test using both simulated and real data several increasingly sophisticated optimization based approaches to restaurant revenue management (RRM) that *address trade-offs between expected revenue, average waiting time and perceived fairness, i.e., the degree of violation of the FCFS rule.*

Contributions

Our contributions are:

1. As an exact dynamic programming approach to RRM is infeasible due to "the curse of dimensionality," we develop mathematical programming based approaches of increasing sophistication in order to maximize expected revenue, yet taking into account expected waiting time and associated fairness issues. We develop integer programming, stochastic programming, and approximate dynamic programming models to provide efficiently implementable policies for RRM. These approaches are interesting in their own right and may find application in revenue management in other contexts. A particularly promising method is the approach of approximate dynamic programming based on an integer programming model.

2. We propose a stochastic gradient algorithm motivated by the work by Karaesmen and van Ryzin [3] to address RRM with reservations. We also apply approximate dynamic programming to make optimal seating decisions for both reservation and walk-in customers online.

3. In computational studies involving both simulated and real data, we show that our models improve revenue often substantially relative to the industry standard of seating parties in a FCFS manner without increasing the average waiting time.

Structure

The structure of the paper is as follows: Section 2 describes the structure and components of the basic model. Section 2.1 describes the integer programming approach, while Section 2.2 outlines the simulation model. Section 3 elaborates on extensions to the basic model. Two variations of the model are introduced in this section: Section 3.1 describes the stochastic programming version of the basic model; Section 3.2 introduces an approximate dynamic programming approach for the basic integer programming model; Section 3.3 describes first-come-first-serve models and the bid pricing model used as a performance baseline for the previous models. Section 4 introduces reservations to our previous models. Section 4.1 describes the reservation booking model that determines the optimal booking level using a stochastic gradient algorithm, while Section 4.2 describes a modified version of the approximate dynamic programming model introduced earlier. In Sections 5 and 6 we report computational results for models without and with reservations, respectively. Section 7 summarizes our findings and contains some concluding remarks.

2 The Basic Model

2.1 An Integer Programming Approach

In this section, we develop an integer programming (IP) model that aims to maximize the expected future revenue, while controlling for expected waiting time by deciding when and where to seat each incoming party. We use a discrete time horizon of N equal length periods so that the number of decision variables and constraints remain tractable. We first introduce the data on which the model is based.

Data

We consider a restaurant that can seat parties of size $k = 2, 4, \ldots, K$ with K even. For simplicity, parties of sizes $1, 3, \ldots, K-1$ can be considered to be one person larger. There are $k' = 2, 4, \ldots, K$ different table sizes such that a party of size k can sit at a table of size k', for $k' \geq k$. We assume that the total revenue generated by a party increases with the party size. Note that we do not necessarily assume that the revenue per person is affected by party size. We model the service time as a collection of SP service phases (for example: first course, second course, dessert). By breaking up the service time and keeping track of how many parties are in each phase, we can make a better estimate of the remaining service time. Our model only uses information about expected values. In this respect, our model is similar to network airline revenue management systems that use a linear programming model (for example the widely used bid price control approach) to decide seat allocation. We assume the following information is known:

c_k	:=	the number of tables of size k available.
$D_{t,k}$	:=	the expected number of size k parties arriving at time t.
$\mu_{k,s}$	:=	the expected duration of phase s for a size k party.
S_k	:=	$\sum_{s=1}^{SP} \mu_{k,s}$, the expected total service duration of a size k party.
$\mu'_{k,s}$	:=	$\sum_{n=s}^{SP} \mu_{k,n}$, the expected remaining service duration for a size k party entering phase s.
R_k	:=	the revenue expected from a party of size k.
$CostQ_{t,k}$	:=	the cost of postponing service to a party of size k that arrived at time t and currently in queue.

$CostX_{t,k}$:= the cost of postponing service to a party of size k that is expected to arrive at time t.

The above data can be collected by many POS softwares. For example, $D_{t,k}$ and R_k can be estimated using historical data on customer arrivals and bills, which is often kept track by POS softwares. We have found there are several POS products that have the ability to track service phases of customers with an electronic ordering pad used by waiters when taking orders. Currently, these pads are used to send orders to the kitchen, but we believe it can also be used to notify the floor manager on the progress of the customers in their meals. With this added feature, it would be easy for the POS software to estimate average service phase durations. Floor managers would need to record arrival and departure times of parties, but that can be readily linked to seating the parties and processing their bills, respectively. Finally, we set the values for $CostQ_{t,k}$ and $CostX_{t,k}$ in our computational experiments (see Section 5) in order to capture the trade-off between quality of service (excessive waiting times and denied service) and revenue.

In addition to revenue, important considerations in managing restaurants are the control of waiting time and associated fairness issues. Thus, in the IP model we explicitly address these issues. For this purpose, we introduce the following parameters that we adjust in order to achieve the "right" trade-off between revenue, waiting time, and fairness.

Max := the maximum number of periods that a party will wait.

M := a user-defined parameter that controls the trade-off between revenue and waiting time (the higher M, the higher the importance of the waiting time; See Eq. (1)).

η := a user-defined parameter that controls the trade-off between revenue and allowing flexibility in allocating a size k party to a table of size $k' \geq k$; See Eq. (1).

In practice, the waiting time behavior of customers are complex. The amount of time a party may wait in queue may follow a probabilistic distribution and may be dependent on the size of the party and time of arrival. However, to simplify the model, we assume that every party is willing to wait up to Max periods. If they are not seated within Max periods, the model assumes that they automatically renege from the queue. It further assumes that parties can not renege earlier nor later than Max periods.

State

The state of the system is described as follows.

now := the current time (in periods).

$Q_{t,k}$:= the number of size k parties that arrived at time t currently waiting in queue.

$N^s_{k,k'}$:= the number of size k parties in service phase s seated at a size k' table.

To better utilize capacity, the IP model allows parties of size k to seat in a table of size $k' \geq k$. The service state parameters $N^s_{k,k'}$ keep track of these parties.

Decision Variables

We introduce the following decision variables:

$q_{t,t',k,k'}$:= the number of parties of size k, who arrived at time t and currently in the queue, that should be seated at a size k' table at time t'.

$qdeny_{t,k}$:= the number of party of size k, who arrived at time t and currently in the queue, that are not currently allocated a seat (i.e., not seated within Max periods).

$x_{t,t',k,k'}$:= the number of parties of size k, expected to arrive at time t, that should be seated at a size k' table at time t'.

$xdeny_{t,k}$:= the number of parties of size k, expected to arrive at time t, that are not currently allocated a seat (i.e. not seated within Max periods).

$z^q_{t,k}$:= the auxiliary variables that allows us to model fairness in the seating decisions, captured in Eqs. (5) and (6) below.

z^x_k := the auxiliary variables that allows fairness in the seating decisions, captured in Eqs. (7) and (8) below.

Each time the IP model is formulated, there are two disjoint sets of customers that it must consider; those that have already arrived and those that are expected to arrive. $q_{t,t',k,k'}$ and $qdeny_{t,k}$ are seating decision variables corresponding to the former type of customers and $x_{t,t',k,k'}$ and $xdeny_{t,k}$ are the seating decision variables for the latter type. The parties represented by the $q_{t,t',k,k'}$'s and $qdeny_{t,k}$'s are parties that have already arrived and thus known. However, the parties represented by $x_{t,t',k,k'}$'s and $xdeny_{t,k}$'s have not yet arrived and are thus uncertain. Partitioning the decision variables in such a way allows for more dynamic and flexible modeling.

The model does not explicitly deny service to the customers, but it may not be able to seat some parties within Max periods - the maximum number of periods that a party would be willing to wait in queue. These parties that are not currently allocated a table is captured by the decision variables $qdeny_{t,k}$ and $xdeny_{t,k}$. The use of the auxiliary variables $z^q_{t,k}$ and z^x_k will be described in more detail in the IP formulation.

The Integer Program Formulation

We formulate and solve an integer program at each customer arrival and departure, and determine whether a party should be seated from the optimal solution. Our objective is to maximize expected revenue while maintaining reasonable waiting times and fairness in our seating decisions, by considering the state of the restaurant. We can also maximize occupancy, by assuming that the revenue per person is independent of the number in the party. Note that even though we solve the IP to optimality, the IP model only provides a tractable approximation to the true stochastic and dynamic problem. We next describe the proposed model.

Objective Function

The objective is the maximization of expected future revenue while controlling waiting time.

$$\max \sum_{\substack{t=1,\ldots,\lfloor now \rfloor \\ k=2,\ldots,K}} \sum_{\substack{k'=k,\ldots,K \\ t'=\lfloor now \rfloor,\ldots,min(N,t+Max-1)}} (R_k - M(t'-t) - \eta(k'-k))q_{t,t',k,k'} - \sum_{\substack{t=1,\ldots,\lfloor now \rfloor \\ k=2,\ldots,K}} CostQ_{t,k} qdeny_{t,k}$$

$$\sum_{\substack{t=\lfloor now \rfloor,\ldots,N \\ k=2,\ldots,K}} \sum_{\substack{k'=k,\ldots,K \\ t'=t,\ldots,\min(N,t+Max-1)}} (R_k - M(t'-t) - \eta(k'-k))x_{t,t',k,k'} - \sum_{\substack{t=\lfloor now \rfloor,\ldots,N \\ k=2,\ldots,K}} CostX_{t,k} xdeny_{t,k} \quad (1)$$

The first set of summations corresponds to the expected revenue that can be attained from parties *currently* in the queue. The second set corresponds to the cost of not allocating a table to parties currently in the queue. The third set corresponds to the revenue from parties *expected* to arrive in the future and the last is the cost of not allocating a table to those customers.

The term $-M(t'-t)$ in the first and third coefficients is a cost again excessive waiting time. The cost $-\eta(k'-k)$ encourages seating from the smallest available table first. Without this term, the model may, for example, seat a party of size two at a table of size eight if it does not expect larger sized parties in the near future. However, since our expected demand data are only estimations and the actual demand in uncertain, we want to be on the safe side and seat from the smallest available table first.

The costs $CostQ_{t,k}$ and $CostX_{t,k}$ are necessary for the model to give preference to customers in the queue as well as those expected to arrive in the current period over those expected to arrive in the future. Thus, the value of $CostQ_{t,k}$ is higher than $CostX_{t,k}$ for all t for each k, because postponing service to a customer already in queue should be higher than postponing service to a customer expected to arrive in the future. Also, $CostX_{t,k}$ for $t = \lfloor now \rfloor$ is higher than that for $t > \lfloor now \rfloor$. If it were the same, the model will consider postponing service to a party that arrives in the current period versus seating that party and postponing service to an equal sized future party as the same. We prefer the former solution, since there is a possibility that a party will arrive in the current period and thus we want the model to give us the ability to seat them. The exact numerical value of these weights are determined by computational experimentation.

Constraints

(1) Demand Constraints:

$$\sum_{\substack{t'=\lfloor now \rfloor,\ldots,min(N,t+Max-1)\\ k'=k,\ldots,K}} q_{t,t',k,k'} + qdeny_{t,k} = Q_{t,k}, \qquad \forall \quad t = 1,\ldots,\lfloor now \rfloor; k = 2,\ldots,K. \tag{2}$$

$$\sum_{\substack{t'=t,\ldots,min(N,t+Max-1)\\ k'=k,\ldots,K}} x_{t,t',k,k'} + xdeny_{t,k} = D_{t,k}, \qquad \forall \quad t = \lfloor now \rfloor,\ldots,N; k = 2,\ldots,K. \tag{3}$$

Constraints (2) insure that the model does not seat more parties than those currently in the queue. Constraints (3) insure that the model does not seat more parties than currently expected.

(2) Seating Capacity Constraints:

This constraint characterizes the capacity constraint at each time τ for each table size k'. It also incorporates a nesting concept that allows parties of size k to sit at a size k' table, where $k \leq k'$.

$$\sum_{\substack{k=2,\ldots,k'\\ t'\in T(t,k)}} \left(\sum_{t=max(1,t'-Max+1),\ldots,\lfloor now \rfloor} q_{t,t',k,k'} + \sum_{t=max(\lfloor now \rfloor,t'-Max+1),\ldots,t'} x_{t,t',k,k'} \right)$$

$$+ \sum_{\substack{k=2,\ldots,k'\\ s=1,\ldots,SP}} I_N(\tau) \leq c_{k'}, \qquad \forall \quad \tau = \lfloor now \rfloor,\ldots,N; k' = 2,\ldots,K. \tag{4}$$

where

$$T(t,k) = \{t' : t \leq t' \leq \tau \leq t' + S_k, \quad t' \in \mathcal{Z}\}, \quad I_N(\tau) = \begin{cases} N^s_{k,k'}, & \text{if } \tau \leq now + \mu'_{k,s}, \\ 0, & \text{otherwise.} \end{cases}$$

Note that the parameters $N^s_{k,k'}$ allow for a more accurate estimation of the remaining service time of the parties currently being served. The last summation term in Eq. (4) corresponds to the expected number of parties currently seated at table k' who are still being served at time τ.

(3) Fairness Constraints:

The model uses parameter M to control waiting time in the objective function. However, when this cost is added, the model would rather seat last-come-first-serve within the same party size because the last customer to arrive would have less waiting time, and thus a higher objective value. To avoid this problem, we add constraints in the model that would seat the customers *within* the same party size in the order that they arrived. By seating parties in queue by first-come-first-serve within the same party size, the waiting time of the parties decrease and fewer parties will leave from the queue. In particular, because of perceived unfairness, customers in a restaurant would strongly complain if a same sized party that arrived after them were seated before them.

The following constraint insures a FCFS seating policy within the same party size for those in the queue that are able to be seated at the current time. It is not extended to future periods due to the uncertainty of the future state of the queue.

$$\sum_{\substack{\tilde{t}=max(1,\lfloor now \rfloor - Max+1),\ldots,t-1 \\ \tilde{k}=k,\ldots,K}} \left(Q_{\tilde{t},k} - q_{\tilde{t},\lfloor now \rfloor,k,\tilde{k}}\right) \leq \left(\sum_{\tilde{t}=max(1,\lfloor now \rfloor - Max+1),\ldots,t-1} Q_{\tilde{t},k}\right) z^q_{t,k}, \quad (5)$$

$$\sum_{\tilde{k}=k,\ldots,K} q_{t,\lfloor now \rfloor,k,\tilde{k}} \leq L(1 - z^q_{t,k}), \quad (6)$$

$$\text{for} \quad t = max(2, \lfloor now \rfloor - Max + 2), \ldots, \lfloor now \rfloor; k = 2, \ldots, K.$$

$$\sum_{\substack{\tilde{t}=max(1,\lfloor now \rfloor - Max+1),\ldots,\lfloor now \rfloor \\ \tilde{k}=k,\ldots,K}} \left(Q_{\tilde{t},k} - q_{\tilde{t},\lfloor now \rfloor,k,\tilde{k}}\right) \leq \left(\sum_{\tilde{t}=max(1,\lfloor now \rfloor - Max+1),\ldots,t-1} Q_{\tilde{t},k}\right) z^x_k, \quad (7)$$

$$\sum_{\tilde{k}=k,\ldots,K} x_{\lfloor now \rfloor,\lfloor now \rfloor,k,\tilde{k}} \leq L(1 - z^x_k), \quad (8)$$

$$\text{for } k = 2, \ldots, K.$$

Here $z^q_{t,k}$, z^x_k are auxiliary binary decision variables associated with each set of constraints. L is some large positive constant. Eqs. (5) and (6) state that if there are parties of size k that arrived before time t still in queue (i.e., the LHS of (5) is positive), then $z^q_{t,k}$ has to be equal to 1. This implies that the LHS of (6) must be zero. In other words, Eqs. (5) and (6) insure that a party of size k that arrived at time t can be seated only when there are no parties of k that arrived before them still in queue. Constraints (7) and (8) are analogous to parties of size k expected to arrive at the current period.

(4) Integrality Constraints:

For $t = 1, \ldots, N; t' = t, \ldots, N; k = 2, \ldots, K; k' = k, \ldots, K$

$$q_{t,t',k,k'} \in \mathcal{Z}^+, \qquad x_{t,t',k,k'} \in \mathcal{Z}^+, \qquad z^q_{t,k} \in \{0,1\}, \qquad z^x_k \in \{0,1\} \tag{9}$$

2.2 The Simulation

The IP described in Eqs.(1)-(9) will be formulated and solved whenever the floor manager needs to make a seating decision. When incorporated into a POS product, the system would optimize the IP model with the current system data each time a new customer arrives or when a customer exits. If the optimal solution of the model indicates a party should be seated or not seated at that moment, the result is conveyed to the floor manager who can act accordingly.

The implementation of the model is simple. There are very few or no added work for the floor manager who uses most POS products. He or she needs only to input the size of a newly arrived party and take note when a party exits. All the information needed to run the model is already being collected in most large restaurants.

To summarize, there are three events that drive the model; (1) customer arrivals, (2) service completion and (3) customer attrition from the queue. The model works in the following way after each of these events:

1. Customer Arrives:
   - Input the party size and time of arrival.
   - Update appropriate $D_{t,k}$.
   - Formulate and solve the IP model using current system data.
   - Act according to the IP solution and update system data, i.e.,
     - If the party is to be seated, seat them at the specified table. Update $N^s_{k,k'}$ accordingly.
     - If the party is not to be seated, put them at the end of the line. Update $Q_{t,k}$ accordingly.
2. Customer Exits After Service Completion:
   - Exit the party from the table.
   - Update appropriate $N^s_{k,k'}$.
   - Formulate and solve the IP model using current system data.
   - Act according to the IP output and update system data, i.e.,
     - If a party currently in the queue is to be seated, take them off the queue and seat them at the specified table. Update $N^s_{k,k'}$ and $Q_{t,k}$ accordingly.

3. Customer Attrition from the Queue

   - Update appropriate $Q_{t,k}$.

We update $N^s_{k,k'}$ and $Q_{t,k}$ by incrementing or decrementing the appropriate term. We do the same for $D_{t,k}$, the expected future demand, instead of using any probabilistic approach to forecasting demand. The initial value of $D_{t,k}$ is set to the expected demand for each time period t and party size k calculated by a simple average of the historical data. As arrivals are seen for a given t and k, $D_{t,k}$ is decremented. If $D_{t,k}$ is 0 when an unexpected customer arrives, the simulation model automatically puts them in the queue, the queue state is updated and the IP is optimized. Although the party is theoretically put in the queue from the model's perspective, they often do not have to wait to be seated. Though this is a simplified estimate of future demand levels, it seems sufficient from our computational experimentations.

3 Extensions Of The Basic Model

In this section, we present several increasingly more sophisticated extensions of the basic model.

3.1 A Stochastic Integer Programming Model

One of the shortcomings of the basic model is its sole use of the expected demand values as an indication of future customer arrivals. The stochastic version of the IP introduces various demand scenarios that the IP model can work with. Several demand scenarios are generated similarly to the process of generating arrivals in simulations. By using the generated scenarios with the expected demand values, the IP attempts to more accurately capture the future demand characteristics.

The stochastic IP model works with Ω demand scenarios (one of them being the expected case), each with probability α_ω, where $\omega = 1, 2, \ldots, \Omega$. The objective function is modified to maximize the expected revenue over the possible scenarios.

Additional and Modified Parameters and Variables

To summarize the new parameters and the modified decision variables:

$x_{t,t',k,k',\omega}$:= the number of size k parties expected to arrive at time t that should be seated at a table of size k' at time t' in scenario ω.

$xdeny_{t,k,\omega}$:= the number of size k parties expected to arrive at time t that are not allocated a table in scenario ω.

Ω := the total number of scenarios.

α_ω := the probability of scenario ω.

$D^{\omega}_{t,k}$:= the number of currently expected size k parties arriving at time t in scenario ω.

$z^{x}_{k,\omega}$:= the auxiliary variables that allows fairness in the seating decision, captured in Eqs. (14) and (15).

Stochastic Integer Programming Formulation

Objective Function

$$\max \sum_{\substack{t=1,\ldots,\lfloor now\rfloor \\ k=2,\ldots,K}} \sum_{\substack{k'=k,\ldots,K \\ t'=\lfloor now\rfloor,\ldots,min(N,t+Max-1)}} (R_k - M(t'-t) - \eta(k'-k))q_{t,t',k,k'} - \sum_{\substack{t=1,\ldots,\lfloor now\rfloor \\ k=2,\ldots,K}} CostQ_{t,k}qdeny_{t,k}$$

$$\sum_{\substack{t=\lfloor now\rfloor,\ldots,N \\ k=2,\ldots,K}} \sum_{\substack{k'=k,\ldots,K \\ t'=t,\ldots,\min(N,t+Max-1) \\ \omega=1,\ldots,\Omega}} \alpha_\omega(R_k - M(t'-t) - \eta(k'-k))x_{t,t',k,k',\omega} - \sum_{\substack{t=\lfloor now\rfloor,\ldots,N \\ k=2,\ldots,K \\ \omega=1,\ldots,\Omega}} \alpha_\omega CostX_{t,k}xdeny_{t,k,\omega} \tag{10}$$

Constraints

(1) Demand Constraints:

The first set of constraints insures that the IP model does not seat more customers than those in the queue. The second set insures that the IP model does not seat more parties than the demand for each scenario.

$$\sum_{\substack{t'=\lfloor now\rfloor,\ldots,min(N,t+Max-1) \\ k'=k,\ldots,K}} q_{t,t',k,k'} + qdeny_{t,k} = Q_{t,k}, \quad \forall \quad t=1,\ldots,\lfloor now\rfloor; k=2,\ldots,K. \tag{11}$$

$$\sum_{\substack{t'=t,\ldots,min(N,t+Max-1) \\ k'=k,\ldots,K}} x_{t,t',k,k',\omega} + xdeny_{t,k,\omega} = D^{\omega}_{t,k}, \quad \forall \quad t=\lfloor now\rfloor,\ldots,N; k=2,\ldots,K; \omega=1,\ldots,\Omega. \tag{12}$$

(2) Seating Capacity Constraint:

$$\sum_{\substack{k=2,\ldots,k' \\ t'\in T(t,k)}} \left(\sum_{t=max(1,t'-Max+1),\ldots,\lfloor now\rfloor} q_{t,t',k,k'} + \sum_{t=max(\lfloor now\rfloor,t'-Max+1),\ldots,t'} x_{t,t',k,k',\omega} \right) + \sum_{\substack{k=2,\ldots,k' \\ s=1,\ldots SP}} I_N(\tau) \le c_{k'},$$

$$\forall \quad \tau=\lfloor now\rfloor,\ldots,N; k'=2,\ldots,K; \omega=1,\ldots,\Omega. \tag{13}$$

where $T(t,k')$ and $I_N(\tau)$ are as in (4).

(3) Fairness Constraint:

Same as (5) and (6). In addition,

$$\sum_{\substack{\tilde{t}=max(1,\lfloor now\rfloor-Max+1),\ldots,\lfloor now\rfloor \\ \tilde{k}=l,\ldots,K}} \left(Q_{\tilde{t},k}-q_{\tilde{t},\lfloor now\rfloor,k,\tilde{k}}\right) \leq \Big(\sum_{\tilde{t}=max(1,\lfloor now\rfloor-Max+1),\ldots,t-1} Q_{\tilde{t},k}\Big) z^x_{k,\omega}, \tag{14}$$

$$\sum_{\tilde{k}=k,\ldots,K} x_{\lfloor now\rfloor,\lfloor now\rfloor,k,\tilde{k}} \leq L(1-z^x_{k,\omega}), \tag{15}$$

$$\text{for } k=2,\ldots K,\ \omega=1,\ldots\Omega.$$

(4) Integrality Constraints:

For $t=1,\ldots,N; t'=t,\ldots,N; k=2,\ldots,K; k'=k,\ldots,K; \omega=1,\ldots\Omega$,

$$q_{t,t',k,k'} \in \mathcal{Z}^+ \qquad x_{t,t',k,k',\omega} \in \mathcal{Z}^+, \qquad z^q_{t,k} \in \{0,1\} \qquad z^x_{k,\omega} \in \{0,1\} \tag{16}$$

The Simulation for the Stochastic Model

The simulation using the stochastic IP model is similar to that of the basic model. The model follows the procedure described in Section 2.2 using the stochastic programming model instead of the deterministic IP model.

3.2 An Approximate Dynamic Programming Model

The approximate dynamic programming model solves for the maximum revenue producing seating policy for each customer. We formulate and solve the IP under each possible seating decision for a particular customer, and the decision that results in the maximum revenue value is chosen.

Let S be a vector describing the current states of the system that can be affected by a seating decision. In particular,

$$S = (Q_{t,k}, N^s_{k,k'}), \qquad \text{for} \tag{17}$$

$$t=\max\{1,\lfloor now\rfloor - Max+1\},\ldots,\lfloor now\rfloor; k=2,\ldots,K; k'=k,\ldots,K; s=1,\ldots,SP.$$

A decision time is when a party arrives or a party exits after service completion. If an arrival of size k occurs, the following decisions are available:

1. Do not seat the incoming party. The new state is such that: $Q_{t,k} \leftarrow Q_{t,k}+1$.

2. For $k' = k, \ldots, K$, seat incoming party at a table of size k'. The new state is such that: $N^1_{k,k'} \leftarrow N^1_{k,k'} + 1$.

If a party of size k occupying a table of size k' exits from the restaurant, the following decisions are available

1. Do not assign the table to a party from the queue. The new state is such that: $N^{SP}_{k,k'} \leftarrow N^{SP}_{k,k'} - 1$.

2. Assign the table to a party of size $k'' \leq k'$, who has been in the queue since time t''. The new state is such that: $Q_{t'',k''} \leftarrow Q_{t'',k''} - 1$, $N^1_{k'',k'} \leftarrow N^1_{k'',k'} + 1$, $N^{SP}_{k,k'} \leftarrow N^{SP}_{k,k'} - 1$.

Let u be the decision taken, so that

$$u = \begin{cases} 0, & \text{Don't seat now,} \\ k', & \text{Seat at a table of size } k' \end{cases} \tag{18}$$

for each of the events. Let S_u be the updated state after a decision u is taken. Let $IP(S)$ be the expected future revenue resulting from the IP model (Eqs. (1) - (9)) as a function of the state S.

Under an approximate dynamic programming (ADP) framework, we choose the decision u, corresponding to seating a size k party that arrived at time t, by solving:

$$\max \left\{ \max_{\substack{u=k,\ldots,K \\ \text{table of size } u \text{ available}}} [R_k - \eta(u-k) + IP(S_u)], \quad IP(S_0) \right\} \tag{19}$$

The first term in the maximization corresponds to the expected revenue of seating the party minus a nesting cost, and the second term corresponds to the expected revenue of not seating the party. This method uses the ADP methodology by approximating the true value function (in a dynamic programming sense) by the value of the IP model.

We note that the IP in (19) need not include the fairness constraints (5)-(8) required in the previous models because ADP evaluates the value function of each party in the order of their arrival and thus ensures FCFS seating within the same party size.

3.3 Comparison Models

First-Come-First-Serve

We develop three first-come-first-serve models to compare their revenue with the revenue generated by using the above models. The FCFS model seats the customers in the order of arrival if there are tables available for them. The "Full Nesting" FCFS Model incorporates full nesting of capacity, where a party of size k is allowed to sit at a size k' table for $k' \geq k$. If there are several possible table sizes, the model would seat them at the smallest of those tables. The "1 Up Nesting" FCFS Model allows a party of size k to sit at either a table of size k or at the next largest table if size k tables are saturated. The "No Nesting" FCFS Model allows no nesting. If there are no tables for a particular customer of size k, but there are tables for smaller customers that arrived after them, the model will seat those customers in order of arrival. Customers who have waited longer than Max will automatically leave the queue. The average of the simulated revenues over several iterations are compared to the revenue of the IPs.

Bid Pricing Model

The bid pricing heuristic commonly applied in airline revenue management is run as a performance benchmark. The linear programming relaxation of the basic IP model is solved and the seating decisions are made based on the difference between the immediate revenue and the sum of the dual prices corresponding to the utilized capacity corresponding to the party's stay. Suppose $u_{t,k'}$ is the dual value of the seating capacity constraint corresponding to table of size k' at time t. The bid price for seating a party of size k to a table of size $k' \geq k$ is as follows:

$$R_k - \sum_{t=\lceil now \rceil}^{\lfloor exit \rfloor - 1} u_{t,k'} - (\lceil now \rceil - now) u_{\lfloor now \rfloor, k'} - (exit - \lfloor exit \rfloor) u_{\lfloor exit \rfloor, k'} \tag{20}$$

where $exit = now + S_k$. The term $(\lceil now \rceil - now) u_{\lfloor now \rfloor, k'}$ and $(exit - \lfloor exit \rfloor) u_{\lfloor exit \rfloor, k'}$ are to prevent over-estimation of the expected stay of the customer due to the discreteness of time. $(\lceil now \rceil - now)$ is the fraction of period $\lfloor now \rfloor$ and $(exit - \lfloor exit \rfloor)$ is the fraction of period $\lfloor exit \rfloor$ that the party is expected to be in service. If there are positive bid prices, the party is seated at the table corresponding to the maximum price. Otherwise, the party is not seated.

4 Models With Reservations

The models in the previous two sections assume that the restaurant does not accept any reservations. Many high-end restaurants, however, do indeed accept reservations for their customers' convenience. Such a policy introduces several decisions which need to be addressed. For example, how much should they over-book and how should they service walk-in customers? We will incorporate reservations in our previous models by using two models: (1) A static reservation booking (RB) model and (2) a dynamic seat allocation model with reservations (DSAR). The static reservation booking model is an off-line model that is optimized using a stochastic gradient approach as in Karaesmen and van Ryzin [3]. The dynamic seat allocation with the reservation model is an online model solved using an ADP approach as in Section 3.2.

4.1 A Reservation Booking Model

In this section, we develop a model that determines how many reservations to accept in prior days for a particular day in the future, given data regarding the expected number of reservation requests, the expected number of walk-ins and the rates for no-shows.

As before, we assume that there are N time periods in which reservations can be accepted. We do not, however, consider nesting of table sizes, i.e., a party of size k can only be assigned to a table of size k. Thus, the seating decisions are independent across the party sizes so that the problem can be broken down into smaller sub-problems for each party size. Since we are building a planning, as opposed to operational, model in this section, we feel that this is justified. Methodologically, motivated by the work of Karaesmen and van Ryzin [3] in a very different context, we solve the optimization problem using a steepest descent algorithm with stochastic gradient estimation.

Data

The following are the data that we expect the restaurant to have for each party size:

R := the expected revenue of the given party size.

p := the probability that a reservation party will show up.

ER_t := the expected number of reservation requests for time t.

EW_t := the expected number of walk-in parties that arrive at time t.

Decision Variables

The key decision is the number of reservations u_t to accept for time period t; $t = 1, \ldots, N$.

Random Variables

We define

z_t := the number of reservation parties booked for time t that show up.

w_t := the number of walk-ins that show up at time t.

We assume that the probability of no-shows are independent and identically distributed across parties of the same size, as shown empirically by Martinez and Sanchez [9]. We further assume that z_t obeys a binomial distribution with parameters $(u_t,\, p)$. We approximate this distribution for time t as a non-homogeneous Poisson distribution of rate pu_t for ease in numerical computation. We also model w_t as a non-homogeneous Poisson with rate EW_t for time t.

Objective Function

We maximize the following objective function $RB(\mathbf{u})$, where $\mathbf{u}$, $\mathbf{ER}$, $\mathbf{z}$ and $\mathbf{w}$ are N-dimensional vectors of elements u_t's, ER_t, z_t's and w_t's, respectively, $t = 1, \ldots, N$.

$$\max_{\mathbf{u} \geq \mathbf{0}} RB(\mathbf{u}) \tag{21}$$

such that

$$RB(\mathbf{u}) = E\left[V(\mathbf{z}, \mathbf{w})\right] - \theta \frac{1}{2} E\left[||\mathbf{u} - \mathbf{ER}||^2\right]. \tag{22}$$

The first term in Eq. (22) corresponds to the expected maximum revenue resulting from having $\mathbf{z}$ reservation customers and $\mathbf{w}$ walk-in customers arrive on the requested date. We will refer to the term $V(\mathbf{z}, \mathbf{w})$ as the the optimal OTD (on-the-day) revenue. This value is calculated by solving a simplified version of the previously discussed seating models of section 2.1-3.2 generalized for reservations. We describe this term in further detail in the following section.

The second term is a regularizing term that discourages the decision variables from straying too far from the expected reservation requests. Without this term, the model may allocate no seats for the 6 o'clock time period and many for the 10 o'clock period, though the expected requests are mostly for 6 o'clock and hardly any for the 10 o'clock period. Thus a policy that allocates significantly more (less) seats for a particular period with few (many) reservation requests overestimates (underestimates) the possible expected revenue. The parameter $\theta > 0$ represents the trade-off between the two terms.

Static On-the-Day Model

We calculate the value of $V(\mathbf{z}, \mathbf{w})$ for a given vector $\mathbf{z}$ and $\mathbf{w}$ by solving an integer program that optimally allocates the tables to reservation and walk-in customers. The OTD model is similar to the previous models with an additional class of reservation parties. However, it does not consider nesting of party sizes and assumes that there are no queues. By making each period large enough (e.g., half an hour), it is reasonable to assume that after one period, both walk-in and reservation parties will renege. Also, most reservation requests are on the half-hour. Thus, this simplification will not significantly jeopardize the accuracy of the model. We will see that these assumptions make the problem a single commodity maximum flow problem which aids us in the convergence analysis of the stochastic gradient algorithm.

The OTD model is also a static model that is not updated dynamically because it is solved before the day in question. It decides off-line how many of the reservation and walk-in parties to seat and to deny service for each time period, $t = 1, \ldots, N$. We need to introduce some additional and modified notation:

Data

S	$:=$	the expected service time for the given party size.
$CostW$	$:=$	the cost of denying service to a walk-in party.
$CostR$	$:=$	the cost of denying service to a party with reservation.
C	$:=$	the total capacity (number of tables) for the given party size.

We assume that the expected revenue and service time distributions are the same for walk-in and reservation customers of the same party size. $CostW$ and $CostR$ are qualitative values of the loss of goodwill and customer dissatisfaction. $CostR$ should be large relative to R, since not being able to serve a customer with a reservation would most likely entail losing that customer and garnering a negative reputation. However, $CostW$ can be negligible since walk-in customers' expectation of being served is significantly less.

Decision Variables

xw_t	$:=$	the number of walk-in parties that are seated at time t.
xr_t	$:=$	the number of reservation parties booked for time t that are seated.
$xwdeny_t$	$:=$	the number of walk-in parties that arrive at time t that are denied service.
$xrdeny_t$	$:=$	the number of reservation parties booked for time t that are denied service.

Problem Formulation

Given a particular vector $\mathbf{z}$ and $\mathbf{w}$, the OTD problem is as follows:

$$\text{(OTD)} \quad V(\mathbf{z},\mathbf{w}) = \max \sum_{t=1,\ldots,N} [R\cdot(xw_t + xr_t) - (CostW)\cdot xwdeny_t - (CostR)\cdot xrdeny_t] \tag{23}$$

$$\begin{aligned}
\text{subject to} \qquad & xw_t + xwdeny_t = w_t && t=1,\ldots,N && (24)\\
& xr_t + xrdeny_t = z_t && t=1,\ldots,N && (25)\\
& \textstyle\sum_{\tau=\max\{1,t-S+1\},\ldots,t}(xw_\tau + xr_\tau) \le C && t=1,\ldots,N && (26)\\
& xw_t \ge 0 && t=1,\ldots,N && (27)\\
& xr_t \ge 0 && t=1,\ldots,N. && (28)
\end{aligned}$$

The objective value is the net revenue of seating xw_t walk-ins and xr_t reservation parties and denying service to $xwdeny_t$ walk-ins and $xrdeny_t$ reservation parties for $t=1,\ldots,N$. Eqs. (24) and (25) are the demand constraints for walk-ins and reservations, respectively, for the particular realizations of demand z_t and w_t for $t = 1, \ldots, N$. Eq. (26) represents the capacity constraint for each time period, $t = 1, \ldots, N$.

Eqs. (23)-(28) can be characterized as a single commodity minimum cost flow problem, thus, the value of $V(\mathbf{z}, \mathbf{w})$ can be calculated by solving the OTD problem as a linear program. It follows by the network structure of OTD and [3] that the function $V(\mathbf{z}, \mathbf{w})$ is submodular and jointly concave with respect to z_t and w_t, $t = 1, \ldots, N$, and $RB(\mathbf{u})$ is component-wise concave, continuous and differentiable with respect to u_t, $t = 1, \ldots, N$. As shown in [3], these properties allow convergence of the stochastic gradient algorithm described in the next section.

Stochastic Gradient Algorithm

We solve the stochastic optimization problem (21) by estimating the stochastic gradient of $RB(\mathbf{u})$ as done in [3]. We apply this gradient to maximize $RB(\mathbf{u})$ using a steepest descent-like algorithm. We first calculate

$$\frac{\partial}{\partial u_t} RB(\mathbf{u}) = \frac{\partial}{\partial u_t} E[V(\mathbf{z}, \mathbf{w})] - \theta(u_t - ER_t), \qquad t = 1, \ldots, N. \tag{29}$$

When $z_t \sim Poisson(pu_t)$ and $w_t \sim Poisson(EW_t)$, $t = 1, \ldots, N$, [3] shows that an unbiased estimate of $\frac{\partial}{\partial u_t} E[V(\mathbf{z}, \mathbf{w})]$ is given by:

$$\frac{\partial}{\partial u_t} E[V(\mathbf{z}, \mathbf{w})] = p(V(\mathbf{z} + \mathbf{e_t}, \mathbf{w}) - V(\mathbf{z}, \mathbf{w})), \qquad t = 1, \ldots, N. \tag{30}$$

The step sizes $\{b_k\}$ satisfy

$$\sum_{k=1}^{\infty} b_k = +\infty, \qquad \sum_{k=1}^{\infty} b_k^2 < \infty. \tag{31}$$

Let $MaxIt$ be the maximum number of iterations we allow. The stochastic gradient algorithm is as follows.

Step 0 Initialization. k=1, $u_t^k = ER_t$ for $t = 1, \ldots, N$.

Step 1 Generate $\nabla RB(\mathbf{u^k})$:

- Generate new vectors $z_t^k \sim Poisson(pu_t^k)$ and $w_t \sim Poisson(EW_t)$ for $t = 1, \ldots, N$.
- Evaluate $V(\mathbf{z^k}, \mathbf{w})$ using a network flow algorithm (for example the network simplex algorithm).
- Evaluate $V(\mathbf{z^k} + \mathbf{e_t}, \mathbf{w})$ for $t = 1, \ldots, N$.
- Derive $\nabla RB(\mathbf{u^k})$ such that, for $t = 1, \ldots, N$:

  $\frac{\partial}{\partial u_t} RB(\mathbf{u^k}) = p(V(\mathbf{z} + \mathbf{e_t}, \mathbf{w}) - V(\mathbf{z}, \mathbf{w})) - \theta(u_t - ER_t)$.

Step 2 Compute

$$u^{k+1} = Project(u^k + b_k \bigtriangledown RB(\mathbf{u^k}))$$

where $Project(\cdot)$ is the projection function to the feasible space $\mathbf{u^k} \geq 0$.

Step 3 If $k > MaxIt$, Exit. Else, set $k := k + 1$ and return to Step 1.

Theorem 1 (Kushner, Clark, Karaesmen, van Ryzin) *Let KT be the set of Kuhn-Tucker points of (21). If KT is a connected set and $\{u^k\}$ is the sequence determined by the previous algorithm with step-sizes satisfying (31), then $\{u^k\} \to KT$ in probability as $k \to \infty$.*

The proof is given Kushner and Clark [8] and [3], Theorem 3 and Theorem 6.3.1, respectively. [8] also shows that a weaker convergence follows when KT is not a connected set.

4.2 Dynamic Seat Allocation with Reservations

The DSAR model is the same as the previous dynamic seating models of Sections 2.1-3.2 but with reservation customers taken into account. It takes the reservations accepted as input and determines the optimal seating policy for each arriving party given the current state of service and queue. It does not, however, update the maximum reservation booking number produced by the RB model. As before, each time the state changes (i.e., due to a customer arrival or customer exit), the state parameters are updated and the model is continually re-optimized online. We apply the ADP approach as in Section 3.2 with a slight modification in the IP formulation.

The following are additional and modified notations for the DSAR model:

Input

$v_{t,k}$:= the number of reservations booked of size k for time t.

$v_{t,k}$ is such that $v_{t,k} \leq u^*_{t,k}$ where $u^*_{t,k}$ is the optimal reservation booking value outputted from the RB model.

Data

$CostQW_{t,k}$:= the cost of postponing service to a walk-in party of size k that arrived at time t and currently waiting in queue.

$CostQR_{t,k}$:= the cost of postponing service to a party of size k with reservations that arrived at time t and currently waiting in queue.

$CostXW_{t,k}$:= the cost of postponing service to a walk-in party of size k that is expected to arrive at time t.

$CostXR_{t,k}$:= the cost of postponing service to a party of size k with reservations that is booked for time t.

p_k := the probability that a party of size k with reservations will show up.

$MaxW$:= the maximum number of periods that a walk-in party will wait.

$MaxR$:= the maximum number of periods that a reservation party will wait.

M_w := a user defined parameter that controls the trade-off between revenue and waiting time for walk-in customers.

M_r := a user defined parameter that controls the trade-off between revenue and waiting time for reservation customers.

The quantities $CostQW_{t,k}$, $CostQR_{t,k}$, $CostXW_{t,k}$ and $CostXR_{t,k}$ are analogous to $CostQ_{t,k}$ and $CostX_{t,k}$ in Section 2.1 which are split into walk-ins and reservation customers (denoted by W and R, respectively). Similarly, $MaxW$ and $MaxR$ are analogous to Max and M_w and M_r are analogous to M in Section 2.1.

State

The following are modified state parameters in addition to those described in Section 2.1.

$Qw_{t,k}$:= the number of size k walk-in parties currently in queue that arrived at time t.

$Qr_{t,k}$:= the number of size k reservation parties currently in queue that arrived at time t.

$Dw_{t,k}$:= the expected number of walk-in parties of size k that are going to arrive in time t who have not already arrived.

$Dr_{t,k}$:= the expected number of reservation parties of size k that are going to arrive in time t who have not already arrived.

When $now < t$, $Dw_{t,k} = EW_{t,k}$ and $Dr_{t,k} = p_k v_{t,k}$. Each time a walk-in or reservation party of size k arrives at time t, $Dw_{t,k}$ or $Dr_{t,k}$ is decremented, respectively.

Decision Variables

$qw_{t,t',k,k'}$:= the number of walk-in parties of size k, that arrived at time t and currently in queue, that should be seat at a size k' table at time t.

$qr_{t,t',k,k'}$:= the number of reservation parties of size k, that arrived at time t and currently in queue, that should be seat at a size k' table at time t.

$qwdeny_{t,k}$:= the number of walk-in parties of size k, that arrived at time t and currently in queue, that are not allocated a table (i.e., are not seated within $MaxW$ periods).

$qrdeny_{t,k}$:= the number of reservation parties of size k, that arrived at time t and currently in queue, that are not allocated a table (i.e., are not seated within $MaxR$ periods).

$xw_{t,t',k,k'}$:= the number of walk-in parties of size k, expected to arrive at time t, that should be seated at a size k' table at time t'.

$xr_{t,t',k,k'}$:= the number of reservation parties of size k, expected to arrive at time t, that should be seated at a size k' table at time t'.

$xwdeny_{t,k}$:= the number of walk-in parties of size k, expected to arrive at time t, that are not allocated a table (i.e., are not seated within $MaxW$ periods).

$xrdeny_{t,k}$:= the number of reservation parties of size k, expected to arrive at time t, that are not allocated a table (i.e., are not seated within $MaxR$ periods).

IP Formulation

DSAR uses a slightly modified version of the ADP model described in Section 3.2 due to its superior performance compared to the other models. The following describes these modifications:

Objective

$$\begin{aligned}
\max \quad & \sum_{\substack{t=1,\dots,\lfloor now \rfloor \\ k=2,\dots,K \\ t'=\lfloor now \rfloor,\dots,\min(N,t+MaxW-1) \\ k'=k,\dots,K}} (R_k - M_w(t'-t) - \eta(k'-k)) qw_{t,t',k,k'} \\
+ & \sum_{\substack{t=1,\dots,\lfloor now \rfloor \\ k=2,\dots,K \\ t'=\lfloor now \rfloor,\dots,\min(N,t+MaxR-1) \\ k'=k,\dots,K}} (R_k - M_r(t'-t) - \eta(k'-k)) qr_{t,t',k,k'} \\
+ & \sum_{\substack{t=\lfloor now \rfloor,\dots,N \\ k=2,\dots,K \\ t'=t,\dots,\min(N,t+MaxW-1) \\ k'=k,\dots,K}} (R_k - M_w(t'-t) - \eta(k'-k)) xw_{t,t',k,k'} \\
+ & \sum_{\substack{t=\lfloor now \rfloor,\dots,N \\ k=2,\dots,K \\ t'=t,\dots,\min(N,t+MaxR-1) \\ k'=k,\dots,K}} (R_k - M_r(t'-t) - \eta(k'-k)) xr_{t,t',k,k'} \\
- & \sum_{\substack{t=1,\dots,\lfloor now \rfloor \\ k=2,\dots,K}} (CostQW_{t,k} \quad qwdeny_{t,k} + CostQR_{t,k} \quad qrdeny_{t,k}) \\
- & \sum_{\substack{t=\lfloor now \rfloor,\dots,N \\ k=2,\dots,K}} (CostXW_{t,k} \quad xwdeny_{t,k} + CostXR_{t,k} \quad xrdeny_{t,k})
\end{aligned} \tag{32}$$

The first four sets of summations correspond to the expected revenue, waiting time cost and nesting costs of seating (1) the walk-in parties currently in queue, (2) the reservation parties currently in queue, (3) the walk-in parties expected to arrive in the future, and (4) the reservation parties expected to arrive in the future, respectively. The last two sets of summations correspond to the cost of postponing service to (1) walk-in parties and reservation parties currently in queue and (2) walk-in parties and reservation parties expected to arrive in the future, respectively.

Constraints

(1) Demand Constraints:

The following constraints are analogous to (2) and (3) in Section 2.1.

$$\sum_{\substack{t'=t,\ldots,min(N,t+MaxW-1)\\k'=k,\ldots,K}} qw_{t,t',k,k'} \quad + \quad qwdeny_{t,k} = Qw_{t,k}, \qquad \forall \quad t=1,\ldots,\lfloor now \rfloor; k=2,\ldots,K. \quad (33)$$

$$\sum_{\substack{t'=t,\ldots,min(N,t+MaxR-1)\\k'=k,\ldots,K}} qr_{t,t',k,k'} \quad + \quad qrdeny_{t,k} = Qr_{t,k}, \qquad \forall \quad t=1,\ldots,\lfloor now \rfloor; k=2,\ldots,K. \quad (34)$$

$$\sum_{\substack{t'=t,\ldots,min(N,t+MaxW-1)\\k'=k,\ldots,K}} xw_{t,t',k,k'} \quad + \quad xwdeny_{t,k} = Dw_{t,k}, \qquad \forall \quad t=\lfloor now \rfloor,\ldots,N; k=2,\ldots,K. \quad (35)$$

$$\sum_{\substack{t'=t,\ldots,min(N,t+MaxR-1)\\k'=k,\ldots,K}} xr_{t,t',k,k'} \quad + \quad xrdeny_{t,k} = Dr_{t,k}, \qquad \forall \quad t=\lfloor now \rfloor,\ldots,N; k=2,\ldots,K. \quad (36)$$

(2) Seating Capacity Constraints:

The following constraint is analogous to the (4) in Section 2.1 with reservation parties.

$$\sum_{\substack{k=2,\ldots,k'\\t'\in T(t,k)}} \left(\sum_{t=\max(1,t'-MaxW+1),\ldots,\lfloor now \rfloor} qw_{t,t',k,k'} + \sum_{t=\max(1,t'-MaxR+1),\ldots,\lfloor now \rfloor} qr_{t,t',k,k'} + \sum_{t=\max(\lfloor now \rfloor,t'-MaxW+1),\ldots,t'} xw_{t,t',k,k'} + \sum_{t=\max(\lfloor now \rfloor,t'-MaxR+1),\ldots,t'} xr_{t,t',k,k'} \right) + \sum_{\substack{k=2,\ldots,k'\\s=1,\ldots,SP}} I_N(\tau) \le c_{k'},$$
$$\forall \quad \tau=\lfloor now \rfloor,\ldots,N; k'=2,\ldots,K. (37)$$

where $T(t,k')$ and $I_N(\tau)$ are as in (4).

(4) Integrality Constraints:

For $t = 1,\ldots,N; t' = t,\ldots,N; k = 2,\ldots,K; k' = k,\ldots,K$,

$$qw_{t,t',k,k'},\quad qr_{t,t',k,k'},\quad xw_{t,t',k,k'},\quad xr_{t,t',k,k'},\quad qwdeny_{t,k},\quad qrdeny_{t,k},\quad xwdeny_{t,k},\quad xrdeny_{t,k} \in \mathcal{Z}^+$$

Incorporating the ADP model

In the DSAR model, the state vector S is characterized by:

$$S = (Qw_{t_w,k}, Qr_{t_r,k}, N^s_{k,k'}), \quad \text{for} \tag{38}$$

$$t_w = \max(1, \lfloor now \rfloor - MaxW + 1),\ldots,\lfloor now \rfloor;$$
$$t_r = \max(1, \lfloor now \rfloor - MaxR + 1),\ldots,\lfloor now \rfloor;$$
$$k = 2,\ldots,K; k' = k,\ldots,K; s = 1,\ldots,SP.$$

The service state parameter $N^s_{k,k'}$ remains the same as in Section 3.2.

There are two types of arrivals: reservation customer arrivals and walk-in arrivals. There is, however, no distinction between reservation and walk-in service completions. Thus, the decision times are when there is a reservation party arrival, walk-in arrival, and a customer service completion. The decisions available at each event are illustrated below:

- Reservation Arrival at time t of size k:
  1. Do not seat the incoming reservation party. The new state is such that: $Qr_{t,k} \leftarrow Qr_{t,k} + 1$.
  2. For $k' = k,\ldots,K$, seat incoming party at a table of size k'. The new state is such that: $N^1_{k,k'} \leftarrow N^1_{k,k'} + 1$.
- Walk-in Arrival at time t of size k:
  1. Do not seat the incoming walk-in party. The new state is such that: $Qw_{t,k} \leftarrow Qw_{t,k} + 1$.
  2. For $k' = k,\ldots,K$, seat incoming party at a table of size k'. The new state is such that: $N^1_{k,k'} \leftarrow N^1_{k,k'} + 1$.
- Party of size k Exits table for k' at time t:
  1. Do not assign the table to a party from the queue. The new state is such that: $N^{SP}_{k,k'} \leftarrow N^{SP}_{k,k'} - 1$.

2. Assign the table to a reservation party of size $k'' \leq k'$, who has been in the queue since time t''. The new state is such that: $Qr_{t'',k''} \leftarrow Qr_{t'',k''} - 1$, $N^1_{k'',k'} \leftarrow N^1_{k'',k'} + 1$, $N^{SP}_{k,k'} \leftarrow N^{SP}_{k,k'} - 1$.

3. Assign the table to a walk-in party of size $k'' \leq k'$, who has been in the queue since time t''. The new state is such that: $Qw_{t'',k''} \leftarrow Qw_{t'',k''} - 1$, $N^1_{k'',k'} \leftarrow N^1_{k'',k'} + 1$, $N^{SP}_{k,k'} \leftarrow N^{SP}_{k,k'} - 1$.

Finally, the DSAR model uses the objective value of the IP described by (32)-(4.2) for $IP(S)$ in (19).

4.3 Comparison Model

First-Come-First-Serve Model with Reservations

We incorporate reservation customers to the FCFS model in Section 3.3 which uses a heuristic to seat walk-in parties within reservations by taking the optimal reservation booking data output from the RB model of Section 4.1. When a walk-in party arrives, the model will check whether seating them would take away a table from any outstanding reservations booked within a 15 minute (1 period) interval of the current time. The three different nesting models (Full Nesting, 1 UP Nesting, and No Nesting) as described in Section 3.3 are tested.

5 Computational Results For Models Without Reservations

In this section, we present the performance of the non-reservation models of Section 2 and 3 on simulated data.

5.1 Data

The test data for the capacity, service time, demand, and revenue are taken from a contrived restaurant. The data is constructed from its dinner time operation, which runs from 6PM to 10PM. We divide this time into 16 equal periods of 15 minute durations.

Capacity Data

This is a small scale restaurant with four tables for two, two tables for four, one table for six and two tables for eight. The restaurant allows nesting of the capacity, i.e., a party of two can be seated at a table of two, four, six or eight.

Service Time Data

The meal duration is split up into three phases: appetizer (phase 1), entrée (phase 2), and coffee and dessert (phase 3) [1]. The expected service durations are illustrated in Table 1. For this example, we have in mind a restaurant with faster than usual turnover time.

	Party Size			
Phase	*2*	*4*	*6*	*8*
1	6	9	12	18
2	39	45	57	75
3	6	6	8	9
Total	51	60	77	102

Table 1: Expected Duration (in minutes) of Service Phases.

Demand Data

The restaurant does not accept reservations. We have tested the various methods for three levels of demand; low, medium and high with average load factors 0.68, 0.93 and 1.54[2]. Each demand level is split up to two demand distributions: constant, where all parties arrive uniformly throughout the day, and varied, where the larger party arrive mainly in the later part of the evening [3]. We simplify the data so that there are only four possible types of customers: parties of size two, four, six and eight. We simulate the customer arrivals as a non-homogeneous Poisson process with rate $\lambda(t, k) :=$ expected demand of size k customers at time t.

Revenue Data

The expected revenue for parties of size two, four, six and eight are \$50, \$120, \$210 and \$320, respectively. We simplify the revenue function so that it is time invariant.

5.2 Algorithms Tested And Parameter Settings

Using the above data, we tested the following algorithms: the first-come-first-serve models (FullNest, 1Up and NoNest), the bid pricing model (BidP), the basic integer programming model (IP), the stochastic programming model using three scenarios (STOCH) and the approximate dynamic programming

[1]The probability distribution of the length of each phase for each party size is approximated by a discrete distribution illustrated in an on-line appendix.

[2]The load factor ρ_k corresponding to party size k was calculated as $\rho_k = \frac{\bar{\lambda}_k S(k)}{\bar{c}(k)}$, where $\bar{\lambda}_k :=$ the expected rate of arrival of size k parties, $S(k) :=$ the expected service time of size k customers and $\bar{c}(k) :=$ the average number of tables of size k over all three configurations.

[3]The on-line appendix contains an illustration of the expected demand for each period and party size corresponding to each of the demand scenarios.

model (ADP). We used CPLEX to solve the optimization models. The models run on Dell Pentium II workstation operating under LINUX.

Parameter Settings

For the STOCH model, we used a three scenario model (i.e., $\Omega = 3$) in which one of the scenarios is the expected demand and the other two are randomly generated. We assign the probability of 0.5 to the expected demand scenario and the probability of 0.25 to each of the generated scenarios. We tested models with larger Ω and with variations in the scenario probability, but the significant increase in the computation time was not worth the small increase in the average revenue.

The value of M is set to 5 for all of the models. After running the models for different values of M, $M = 5$ produced the highest revenue on average for most of the models. Setting this value too high makes the model averse to seating customers that have been waiting in the queue for a long time. When M is set to zero, the model would give no consideration to waiting time when solving for the optimal solution. For values of M between 0 and 5, the resulting average revenue actually increases because the model is forced to seat a customer at a table which it was reserving for a future customer. This may be beneficial because the model sometimes incorrectly forecasts future demand, and keeps tables idle when they could have been used.

The appropriate values for η seem to vary according to the model and the demand level. For the IP and STOCH model, we set $\eta = 4$ and for ADP we set $\eta = 0.25$. These values were chosen from empirical testing, thus they may not be the optimal. However, the differences in revenue and average waiting time were not significant for slight changes in η.

The value for Max is set to 3 for all models, thus we are assuming customers do not wait for more than 45 minutes. The models automatically exit customers who have been waiting for more than three periods.

For the IP and STOCH model, the values of $CostQ_{t,k}$ are 2.5 for $k = 2$, 6.0 for $k = 4$, 10.5 for $k = 6$ and 16.0 for $k = 8$, $\forall t$. $CostX_{t,k}$ when $t = \lfloor now \rfloor$ are 0.5 for $k = 2$, 1.2 for $k = 4$, 2.1 for $k = 6$ and 3.2 for $k = 8$, and when $t > \lfloor now \rfloor$ $CostX_{t,k}$ is set to 0 for all k. The ADP model performed better with significantly higher values of $CostQ_{t,k}$ and $CostX_{t,k}$ than IP and STOCH. Thus these costs for ADP were set to 10 times that of IP and STOCH.

Appropriate values for these parameters are clearly restaurant dependent. They would depend on the demand level and characteristics, expected revenue, expected service time, capacity and waiting time behavior of the customers of each restaurant. We suggest testing for the right values for these parameters by simulation, using the data and characteristics of each restaurant.

5.3 Results

Table 2 contains the average daily revenue, average waiting time per customer and the percentage of customers served for all the algorithms we tested. Each sub-table corresponds to a demand scenario, low and constant, low and varied, medium and constant, medium and varied, high and constant, and high and varied. Table 3 contains the average run times per party for each model.

For smaller load factors, Full Nesting is better than the other FCFS models and performs similarly to the IP, in general. This implies that the phenomenon that the IP captures to increase revenue for small load factors is nesting. The No Nesting model loses revenue by unnecessarily saving large tables for large parties. As demand level increases, the nesting decisions become more complex. This explains the decrease in the revenue gap across the FCFS models with higher demand.

We observe that increasing the sophistication of the models results in monotonically increasing revenue. There is a marginal revenue improvement of using stochastic programming versus the deterministic IP. The ADP model outperforms the deterministic and stochastic model in all demand scenarios, with about 2% improvement from FCFS in low and medium load factors and 12% improvement in high load factors in revenue.

We also observe that the models do not sacrifice waiting time for higher revenue. The optimization models had comparable waiting times to the FCFS models, and in most of the larger load cases had lower waiting time than the FCFS. Thus higher revenue was achieved without any sacrifices and some improvements in the average waiting time. When examining the waiting time for each party size separately, we see that the optimization models have significantly lower waiting times for parties of size six and eight while slightly higher for parties of two compared to the FCFS models. Thus, our models have a smaller range of waiting times across party sizes.

In addition, IP and STOCH have a lower percentage of parties served than the best performing FCFS models, while producing higher revenue. **Overall the ADP model seems the best performing method: it serves about the same percentage of parties as the FCFS models, does not increase and occasionally decreases waiting time, and produces significantly higher revenue.** This implies that the optimization models are able to seat more of the "right" (higher revenue producing) customers. The run times are also in a practical range for all of the models. Thus any of these models can be run online with a POS system.

6 Computational Results For Reservation Models

In this section, we report computational results for the models with reservations. We first run the static reservation booking model (RB) to decide a priori how many reservations to accept, and we then run the dynamic seat allocation model with reservations (DSAR).

6.1 Data

The test data was taken from *Soto's*, a Japanese restaurant in Atlanta, Georgia [7]. Similar to the previous data set, the data are taken from its dinner time operations of 16 periods, from 6PM to 10PM. The service time and revenue data are also identical to that of Section 5.1.

Capacity

Soto's has many small tables that can be put together to accommodate large parties. For the purpose of our model, we assume that the restaurant takes only one table configuration. In this configuration, *Soto's* has 16 tables for two, 7 tables for four, three tables for six and one table for eight. We allow nesting of capacity as before.

Demand Data

Soto's gets around a total of 90 customers on weekdays and 120 to 130 customers on weekends. We test the models on these two demand levels, with corresponding loads of 0.93 and 1.24, respectively. Around 30% of these customers are reservation customers with a no show rate of 3 to 15%. We use a constant no show rate of 10%. Out of the walk-in parties, 55% are of size two, 30% are of size four and 15% are of size six. Out of the reservation parties, 40% are of size two, 43 % are of size four, 10 % are of size six and 7% are of size eight. The distribution of reservation customers throughout the day is as follows: 20% during 6PM to 7PM, 40% during 7PM to 8PM, 30% during 8PM to 9PM, and 10% during 9PM to 10PM. The distribution of walk-in customers is as follows: 30% during 6PM to 7PM, 35% during 7PM to 8PM, 25% during 8PM to 9PM, and 10% during 9PM to 10PM.

6.2 Parameter Settings

We use the following values for $CostQW_{t,k}$: 2.5 for $k = 2$, 6.0 for $k = 4$, 10.5 for $k = 6$, and 16.0 for $k = 8$, $\forall t$. The values for $CostQR_{t,k}$ are 150 for $k = 2$, 300 for $k = 4$, 500 for $k = 6$, and 700 for $k = 8$, $\forall t$. Distinction for the values for both $CostXW_{t,k}$ and $CostXR_{t,k}$ for $t = \lfloor now \rfloor$ are set higher than for $t > \lfloor now \rfloor$. This implies that parties expected to arrive in the current period is given more priority that those expected to arrive later because the state of the distant future in more uncertain than the near future. The values for $CostXW_{t,k}$ when $t = \lfloor now \rfloor$ are 0.05 for $k = 2$, 0.12 for $k = 4$, 0.21 for $k = 6$, and 0.32 for $k = 8$. When $t > \lfloor now \rfloor$, the values are set to 0 for all k. The values for $CostXR_{t,k}$ when $t = \lfloor now \rfloor$ are 101 for $k = 2$, 241 for $k = 4$, 421 for $k = 6$, and 641 for $k = 8$. When $t > \lfloor now \rfloor$, the values are 100 for $k = 2$, 240 for $k = 4$, 420 for $k = 6$, and 640 for $k = 8$.

The value of Mr was set to 8 and Mw was set to 3, reflecting higher cost for keeping reservation customers wait. η was set to 0.5. $MaxR$ was set to 4 and $MaxW$ was set to 2, which reflects the customer behavior at *Soto's*.

6.3 Computational Results

The average revenue, percentage of customers served and average waiting time for reservation and walk-in customers are illustrated in Table 4 and Table 5. Table 4 uses the demand data of 90 customers and Table 5 uses the demand data of 120 customers. In the low demand scenario of 90 customers, DSAR out-performs all of the FCFS models by 3.5% to 6.9%. DSAR also serves a larger percentage of both reservation and walk-in customers than the FCFS models. The average wait for reservation parties is slightly higher for DSAR, but 0.27 periods (4.05 minutes) is still a reasonable length of wait. The average wait for walk-in customers are lowest using DSAR. The results for 120 customers are similar. DSAR out-performs the FCFS model by 6.43% to 8.29%. It again has the best percentage seated for both the reservation and walk-in customers. It also has the highest average waiting time (0.40 periods or 6 minutes) for reservation parties and the lowest average waiting time for walk-in parties. Thus, the DSAR produces more revenue and serves more customers than FCFS for both low and high demands. The results also imply that the DSAR has a higher revenue impact with higher demand.

7 Summary And Concluding Remarks

We feel we gained the following insights from the computational study:

1. For models without reservations, optimization based strategies outperform FCFS based strategies for all low and medium load factors and significantly for high load factors. Somewhat surprisingly optimization based strategies do not affect adversely the service quality (waiting times either remain unchanged or decrease somewhat, while FCFS is maintained within parties of the same size).

2. Increasing the sophistication in the models results in higher revenue without sacrificing waiting time. We believe that the performance of the ADP model represents the best trade-off between maximizing revenue and maintaining low average waiting time and run time.

3. The reservation models we propose (using a stochastic gradient approach to decide the reservations a priori, and ADP to implement it on-line) result in a significant improvement relative to the FCFS models for both low and high demand levels. The RB and DSAR models result both in higher revenue and in lower customer attrition.

4. Overall we feel that optimization based models have a role to play in restaurant revenue management.

There are many areas for future research:

1. Extending our model to support dynamic capacity - that is, allow restaurants to move their tables around to better accommodate the demand at each time.

2. Incorporating balking and reneging.

3. Further empirical testing; this might be facilitated by combining algorithms from this paper with on-line restaurant reservation providers.

Acknowledgments

Research supported in part by the Singapore-MIT alliance. We thank Mr. Sotohiro Kosugi, owner and chef of *Soto's*, a Japanese restaurant in Atlanta, Georgia for sharing with us data from his restaurant and helping us better understand restaurant operations. We also thank the reviewers of the paper and the associate editor for many insightful suggestions.

References

[1] Ahuja, Ravindra K.; Magnanti, Thomas L.; Orlin, James B., *Network Flows: Theory, Algorithms, and Applications,* Prentice Hall, NJ, 1993.

[2] Erlenkotter, D. (1970), "Preinvestment Planning for Capacity Expansion: A Multilocation Dynamic Model," Ph.D. Thesis, Stanford University, CA.

[3] Karaesmen, I. and G. van Ryzin (1998), "Overbooking with Substitutable Inventory Classes," Working Paper.

[4] Kimes, Sheryl E.; Chase, Richard B.; Choi, Sunmee; Lee, Philip Y.; Ngonzi, Elizabeth N., "Restaurant Revenue Management: Applying Yield Management to the Restaurant Industry," *Cornell Hotel and Restaurant Administration Quarterly*, Vol. 39 (3), June 1998, p 32-39.

[5] Kimes, Sheryl E., "Implementing Restaurant Revenue Management: A Five Step Approach," *Cornell Hotel and Restaurant Administration Quarterly*, Vol. 40 (3), June 1999, p 16-21.

[6] Kimes, Sheryl E.; Barrash, Deborah I.; Alexander, John E., "Developing a Restaurant Revenue-Management Strategy," *Cornell Hotel and Restaurant Administration Quarterly*, Vol. 40 (5), Oct 1999, p 18-29.

[7] Kosugi, Sotohiro. Owner and manager of *Soto's* in Atlanta, Georgia.

[8] Kushner, H.J.; Clark, D.S., *Stochastic Approximation Methods for Unconstrained Systems*, Springer-Verlag, AMS vol.26, New York, 1978.

[9] Martinez, R. and M. Sanchez (1970), "Automatic Booking Level Control," *Proceedings of the Tenth AGIFORS Symposium*, American Airline, New York.

[10] McGill, Jeffrey I.; Van Ryzin, Garret J., "Revenue Management: Research Overview and Prospects," *Transportation Science*, Vol. 33, No. 2, May 1999, p 233-256.

[11] Muller, Christopher C., "A Simple Measure of Restaurant Efficiency," *Cornell Hotel and Restaurant Administration Quarterly*, Vol. 40 (3), June 1999, p 31-37.

[12] Quain, Bill; Sansbury, Michael; LeBruto, Stephen M., "Revenue Enhancement, Part1: A Straightforward Approach for Making Money," *Cornell Hotel and Restaurant Administration Quarterly*, Vol. 39 (5), Oct 1998, p 41-48.

[13] Sill, Bill; Decker, Robert, "Applying Capacity-management Science," *Cornell Hotel and Restaurant Administration Quarterly*, Vol. 40 (3), June 1999, p 22-30.

[14] Sill, Brian, "Capacity Management: Engineering the Balance Between Customer Satisfaction, Employee Satisfaction and Company Profit," *The Consultants*, Second Quarter 2000, p 72-81.

[15] Vakharia, A.J.; Selim, H.S.; Husted R.R., "Efficient Scheduling of Part-time Employees," *Omega*, Vol. 20 (2), March 1992, p 201-13.

Load=0.68 Constant	*FullNest*	*1Up*	*NoNest*	*BidP*	*IP*	*STOCH*	*ADP*
Revenue ($)	2351.94	2390.32	2125.46	2328.10	2354.46	2375.92	2401.56
% Difference	0.00%	1.63%	-9.63%	-1.01%	0.11%	0.98%	2.11%
Average Wait (min)	5.0	5.3	10.7	6.0	5.7	5.7	4.9
% Served	93.23%	93.08%	83.25%	90.89%	91.81%	91.94%	93.67%

Load=0.68 Varied	*FullNest*	*1Up*	*NoNest*	*BidP*	*IP*	*STOCH*	*ADP*
Revenue ($)	2066.54	2031.18	1883.90	1997.86	2087.14	2077.30	2112.34
%Difference	0.00%	-1.71%	-8.84%	-3.32%	1.00%	0.52%	2.22%
Average Wait (min)	7.7	7.5	13.1	6.0	8.3	7.8	7.5
% Served	88.06%	88.11%	78.52%	90.89%	86.51%	87.35%	88.58%

Load=0.93 Constant	*FullNest*	*1Up*	*NoNest*	*BidP*	*IP*	*STOCH*	*ADP*
Revenue ($)	2964.62	2928.98	2702.36	2822.12	2976.84	2969.50	3034.18
%Difference	0.00%	-1.20%	-8.85%	-4.81%	0.41%	0.16%	2.35%
Average Wait (min)	9.9	9.8	14.4	11.3	11.1	10.4	9.9
% Served	87.82%	87.69%	78.30%	83.35%	85.48%	86.26%	87.84%

Load=0.93 Varied	*FullNest*	*1Up*	*NoNest*	*BidP*	*IP*	*STOCH*	*ADP*
Revenue ($)	2624.94	2596.36	2391.00	2430.18	2610.06	2651.06	2702.71
%Difference	0.00%	-1.09%	-8.91%	-7.42%	-0.57%	1.00%	2.96%
Average Wait (min)	10.8	10.4	15.9	12.6	10.8	11.3	10.5
% Served	83.16%	83.96%	75.35%	78.22%	82.71%	82.08%	83.31%

Load=1.54 Constant	*FullNest*	*1Up*	*NoNest*	*BidP*	*IP*	*STOCH*	*ADP*
Revenue ($)	3409.82	3467.66	3498.8	3203.10	3676.74	3708.76	3857.36
%Difference	0.00%	1.70%	2.61%	-6.06%	7.83%	8.77%	13.13%
Average Wait (min)	12.3	24.2	25.9	23.7	23.7	24.0	22.9
% Served	68.84%	68.70%	60.36%	57.85%	65.65%	64.02%	68.96%

Load=1.54 Varied	*FullNest*	*1Up*	*NoNest*	*BidP*	*IP*	*STOCH*	*ADP*
Revenue ($)	3146.76	3154.98	3051.06	2902.52	3393.36	3387.42	3545.40
%Difference	0.00%	0.26%	-3.04%	-7.76%	7.84%	7.65%	12.67%
Average Wait (min)	24.0	23.7	27.5	24.3	24.0	23.7	23.4
% Served	67.83%	68.50%	57.16%	56.77%	64.25%	64.33%	66.56%

Table 2: Revenue, Average Wait and Percent Served.

	FCFS	*BidP*	*IP*	*STOCH*	*ADP*
Average Run Time (sec)	0.00	0.67	0.21	0.66	1.07

Table 3: Run Time per Party in seconds.

Demand = 90	*FullNest*	*1Up*	*NoNest*	*DSAR*
Revenue ($)	6944.68	6917.18	6777.02	7182.20
%Difference	0.00%	-0.40%	-2.41%	3.42%
% Reservation Served	95.96%	95.95%	96.02%	97.97%
Avg Wait Reservation (min)	3.0	3.2	3.3	4.0
% Walk-in Served	81.19%	80.90%	79.07%	85.02%
Avg Wait Walk-in (min)	11.1	11.3	12.3	8.6

Table 4: Revenue, Percent Served and Average Waiting Time of Reservation Models for Demand Level 90.

Demand = 120	*FullNest*	*1Up*	*NoNest*	*DSAR*
Revenue ($)	8210.70	8274.82	8132.2	8806.60
%Difference	0.00%	0.78%	-0.96%	7.26%
% Reservation Served	94.57%	94.42%	93.92%	97.39%
Avg Wait Reservation (min)	3.6	3.75	4.2	6.0
% Walk-in Served	53.97%	53.75%	50.39%	65.83%
Avg Wait Walk-in (min)	25.4	25.5	26.7	19.5

Table 5: Revenue, Percent Served and Average Waiting Time of Reservation Models for Demand Level 120.