2010

# Optimizing Restaurant Reservation Scheduling

Jacob Feldman
*Harvey Mudd College*

# Optimizing Restaurant Reservation Scheduling

**Jake Feldman**

Susan E. Martonosi, Advisor

John M. Bossert, Reader

May, 2010

HARVEY MUDD
C O L L E G E
Department of Mathematics

# Abstract

We consider a yield-management approach to determine whether a restaurant should accept or reject a pending reservation request. This approach was examined by Bossert (2009), where the decision for each request is evaluated by an approximate dynamic program (ADP) that bases its decision on a realization of future demand. This model only considers assigning requests to their desired time slot. We expand Bossert's ADP model to incorporate an element of flexibility that allows requests to be assigned to a time slot that differs from the customer's initially requested time. To estimate the future seat utilization given a particular decision, a new heuristic is presented which evaluates time-slot/table assignments based on the expected number of unused seats likely to result from a given assignment. When compared against naive seating models, the proposed model produced average gains in seat utilization of 25%.

# Contents

# Chapter 1

# Introduction

The restaurant business is a cut-throat industry where profit margins are thin for most restaurants. Restaurateurs might have savvy business partners who can manage the money, but what they lack is a mathematician's insight into optimizing restaurant operations. Restaurants make numerous day-to-day decisions regarding the scheduling and timing of events. Examples of such decisions include the assignment of customers to specific tables, how quickly to clear tables once diners have finished, and how long a waiter or waitress should wait before delivering the check. Each of these examples is comprised of a set of decisions, and thus techniques from operations research are useful in modeling these problems and finding the most profitable solution.

Bossert (2009) notes that the profit margins of many fine dining establishments are between only one and three percent of their revenue. As a result, it is essential that restaurants operate their business with the utmost efficiency. The ability to run a highly efficient restaurant is dependent on how the restaurant manages two key elements of demand around the current occupancy of the restaurant. The first element of demand that must be managed is represented by the density of walk-in traffic. Walk-in customers are parties that come to the restaurant without a reservation. The restaurant must consider the timing and magnitude of its walk-in traffic. The second element of demand is represented by reservation requests that are called in. Bossert believes that this element of demand is very important for most restaurants that take call-in reservations, as he thinks there is a direct link between reservations and revenue. While most restaurants must manage both elements of demand, there are some that only take walk-in customers and others that only accept reservations. Among the restaurants that only schedule reservations, the restaurants with consistent streams of daily reservation requests generally have the largest revenues.

Research within this field is strongly influenced by the theories of revenue management: the application of information systems and pricing strategies to allocate the right capacity to the right customer at the right place at the right time (Kimes et al., 1998). Netessine and Shumsky (1999) describe five defining features of problems that can be solved from a revenue-management perspective:

1. It is expensive or impossible to store excess resources;
2. Commitments need to be made when future demand is uncertain;
3. The firm can differentiate among customer segments;
4. The same unit of storage can be used to deliver many different products and services; and
5. Producers are profit-oriented and have broad freedom of action.

The restaurant-reservation problem fits the mold perfectly, as each of the five characteristics are satisfied. The resource in this context refers to the occupancy of the seats in the restaurant at a specific time. It is impossible to store excess resources because if a seat is not filled during a specific time, the chance to fill this time-slot/table combination is lost forever. The second feature is relevant because the restaurant must make an accept/reject decision while future demand remains uncertain. The restaurant can never predict exactly how many reservation requests it will receive in the future, nor can it predict the party size of each request. With regards to the third feature, a restaurant is able to differentiate between customer segments based on party sizes. The fourth feature is satisfied because a given seat at a particular time can be assigned to any customer. Finally, the restaurant has a broad freedom of action because it is allowed to reject service to any customer. Such practices would be questionable, for example, in the context of hospital operations. Further, revenue-management techniques are also useful when variable costs (costs to produce the food) are low, and the capacity (number of seats in the restaurant) is fixed. When this is the case, there is a strong positive correlation between revenue and profit and thus both are equal measures of success for a restaurant.

When revenue-management techniques are applied with regards to the restaurant-reservation scheduling problem, the capacity refers to the table size, the customer becomes the party inquiring about a reservation, and the time represents all possible dining hours. Upon the arrival of a reservation request, the restaurant must ask: Is seating this party at this specific table at this time going to maximize revenue? To answer this question accurately, a restaurant must use tools for forecasting future demand as well as for managing a perishable resource.

This thesis provides an analysis of how restaurant reservations can be studied and forecasted to maximize revenue. An approximate dynamic program was developed to decide whether it is more profitable to accept or reject a given pending request. While accepting a pending request leads to immediate realized revenue, it is possible that this acceptance will diminish future seat utilization. Future seat utilization refers simply to how many people the restaurant will seat in the future.

The rest of this paper is structured as follows. Chapter 2 provides a summary of all relevant research and introduces the accept/reject model proposed by Bossert. Chapter 3 details an extension to the accept/reject model; an element of flexibility is incorporated that allows requests to be assigned to a time slot that differs from their initial request. Chapter 4 describes the restaurant reservation simulation process and the results from these simulations. Finally, Chapter 5 offers avenues for future work and provides a concluding summary of the results.

# Chapter 2

# Background on Restaurant Revenue Management

Previous research relevant to the restaurant-reservation scheduling problem is summarized in this section. After this brief literature review, the model presented in Bossert (2009) is outlined. This model forms the basis for the dynamic program proposed later in this thesis.

## 2.1 Literature Review

The problem of managing restaurant reservation requests is only a subfield within a more general field of revenue management. Netessine and Shumsky (1999) note that a variety of industries (hotel, car rental, and airlines) have implemented revenue management techniques in an attempt to answer the overarching question of how a firm should market and distribute goods to multiple customer segments. McGill and Van Ryzin (1999) review the forty-year history of research within the field of revenue management. They cover developments in forecasting, overbooking, seat inventory control, and pricing as they relate to the airline industry. Relevant to the restaurant-reservation scheduling problem, McGill and Van Ryzin believe that the fundamental revenue management decision to be made is whether to accept or reject a booking. They note that a request should be satisfied only if the fare value of the requested itinerary equals or exceeds the expected displacement cost.

Kimes et al. (1998) focus on the general overview of revenue management and how it relates to the restaurant industry. They analyze all aspects of the dining service from table turnover time to meal durations. Instead of using only revenue, they propose that the measure of success for a restaurant should be revenue per available seat hour (RevPASH). This metric is calculated by dividing the total revenue generated over a specific time by the sum of the number of hours each table is available to accommodate customers. Tables that are available for assignment but go unused for prolonged periods of time therefore contribute very little to the RevPASH value. Kimes et al. (1999) analyze the effects of implementing these strategies at a 100-seat restaurant. They develop a three-step process intended to increase revenue. The process involves collecting data, identifying the underlying causes of operational problems, and then developing a revenue-management strategy to make service more efficient.

Thompson (2002) begins his research by looking at the impact of combinability in restaurants that only have walk-in customers . Combinability, for example, refers to the ability of a restaurant to combine two two-tops (two tables that can seat up to two people each) in order to accommodate a four-person party. Thompson and Kimes (2005) build on this idea of optimal combinability by describing how to find the optimal table mix. They use three heuristics based on integer programs and two variants of simulated annealing heuristics to find the optimal mix of two-, four-, six-, and eight-tops at restaurants that have constant demand, and at restaurants that have varying demand based on the day of the week.

Thompson and Kwortnik (2008) examine whether restaurant reservations should be locked to specific tables at the time the reservation is made, or whether the reservations should be pooled and assigned to tables in real time.

The reservation-inventory question that is the focus of this paper is first addressed by Shioda (2002). She develops an integer-programming model that determines how many reservations to accept in days prior to a particular day in the future. While this model is effective in determining the number of reservations to accept, it fails to incorporate any details of the table layout. This model does not assign reservations to specific tables and it does not consider the possibility of reconfiguring the current table layout in an effort to accept more reservations. Vidotto et al. (2007) also investigate the relevant reservation inventory problem but do so in a manner that does not focus primarily on yield-management. They provide a model that can effectively manage a stream of reservation requests, but this model makes no attempt to identify and select the most profitable requests.

Bossert (2009) has also focused on the scheduling of reservations. He uses an approximate dynamic program to determine whether or not to accept a given reservation requests. He does so by implementing Monte

Carlo simulations where the effective worth of both the accept and reject options is evaluated through a two-pass heuristic that manages a theoretical stream of future demand. Within his simulations, he considers various table mixes for restaurants of varying sizes. A major shortcoming of Bossert's model is that it does not consider the profitability of offering a customer a reservation at a time that differs from their initial request. Section 2.2 gives a detailed account of the current model proposed by Bossert, as it forms the foundation for this thesis.

## 2.2 The Dynamic Accept/Reject Model for Reservations

ʻ

This section includes a summary of the model proposed in Bossert (2009). The interested reader can refer to the original paper for details.

The time during which the restaurant is available to accept reservation requests before the given service date is broken into $T$ distinct time periods, which are assumed to be narrow enough that at most one reservation is requested in each period. The restaurant can only receive requests in a period $k$, where $k < T$. During each period $k$ there is a set probability that a customer will call in for a reservation. Dummy reservations of all zeros are placed in the time periods that do not receive reservation requests. The actual meal service is discretized into $M$ slots. A given request, if accepted, will occupy a certain number of slots, this quantity is referred to the meal duration of a party. This model assumes a constant meal duration over all party sizes.

Bossert uses a dynamic programming framework to weigh the benefits of accepting versus rejecting a pending reservation request. Accepting the reservation will lead to immediate revenue, but it is possible that this acceptance will diminish future seat utilization. The restaurant can thus decide to reject a reservation request for two reasons: an assignment can result in unused seats, or an assignment can result in a large reduction of the turn count of a table. Bossert defines the turn count of a table to be the maximum number of requests that could be assigned to a given table throughout the entire service period. To elucidate the notion of turn count, consider a restaurant that only has one two-top and six dining slots. Assume that the meal duration is two. The turn count for the single table is three; ideally, the restaurant would like to assign reservations to slots 1 and 2, 3 and 4, and 5 and 6. If the restaurant begins by accepting a reservation for slots 2 and 3, then requests can only be feasibly assigned to slots 4 and 5 or 5 and 6. This assignment would thus reduce the turn count of this table from three to one.

The accept/reject model makes a few simplifying assumptions with regards to the characteristics of the restaurant. The table mix (number of tables of size two, four, six, etc...) is predetermined. Most restaurants have a set table mix, but do not have the space to use all the available tables at once. Restaurants thus use subsets of the complete set of tables where the union of each of these subsets forms the complete list of tables. The restaurant then switches between these subsets during the dining service in order to accommodate requests of various party sizes. One such subset is referred to as a feasible table configuration. There is a predetermined set of feasible table configurations that is assumed to be smaller than the set of all possible configurations to maintain computational tractability.

### 2.2.1 Approximate Dynamic Programming Model

Bossert (2009) uses an approximate dynamic program where the decision at each stage is whether to accept or reject the pending reservation request. The vector $(n_k, \vec{s_k}, \vec{t_k}, s_k^*, t_k^*)$ details the current state of the restaurant. The first three components of this vector describe the already accepted requests, whereas the latter two correspond to the details of the pending request that is being considered. In his paper, Bossert defines the variables as follows:

$n_k$ : the number of requests accepted over periods 1 to $k-1$;

$\vec{s_k}$ : vector of length $n_k$, corresponds to the party sizes of the accepted requests;

$\vec{t_k}$ : vector of length $n_k$, corresponds to the arrival time slots of the accepted requests;

$s_k^*$ : size of the pending reservation request; and

$t_k^*$ : requested arrival slot of the pending reservation request.

If the pending reservation request is rejected, the restaurant then considers the next request. This new request will have a party size, $s_{k+1}^*$, and an arrival time, $t_{k+1}^*$. Equation 2.1 shows the changes made to the vector when the pending request is rejected. If the pending request is accepted, updates must be made to each variable. Notice that $n_k$ becomes $n_k + 1$ because the restaurant has accepted an additional request. We also must update $\vec{s_k}$, as we must store the party size of the newly accepted request: $\vec{s_k}$ becomes $\vec{s_k} \cup s_k^*$. The arrival-slots vector is updated in a similar way. Finally, a new pending request must be considered with party size $s^*_{k+1}$ and arrival time $t^*_{k+1}$. The dynamics for the accept case are shown in Equation 2.2.

$$(n_k, \vec{s_k}, \vec{t_k}, s^*_k, t^*_k) \to (n_k, \vec{s_k}, \vec{t_k}, s^*_{k+1}, t^*_{k+1}). \tag{2.1}$$

$$(n_k, \vec{s_k}, \vec{t_k}, s^*_k, t^*_k) \to (n_k + 1, \vec{s_k} \cup s^*_k, \vec{t_k} \cup t^*_k, s^*_{k+1}, t^*_{k+1}). \tag{2.2}$$

The decision to accept or reject the pending request is based on a realization of future demand. At each step, a specific reservation request is considered and a preliminary capacity check is made. Since the previously accepted requests are not assigned to specific tables, the capacity calculation involves shifting the accepted requests to accommodate the current pending request. This capacity calculation is approximate due to the fact that every possible arrangement of the accepted requests cannot be considered. A pending request can only be accepted if there is sufficient room to accommodate the pending request as well as all previously accepted requests. If there is not sufficient space, the reservation is rejected. If the restaurant has the potential to accommodate the pending request, then the benefits of accepting versus rejecting this request must be considered. The benefits in this scenario refer to the potential future seat utilization. This future seat utilization will almost always be higher when the reject option is considered, although it is important to remember that both the future seat utilization and the immediate revenue contribute to the value of the accept option.

The dynamic program considers the two options one at a time. First, the pending request is tentatively rejected and the potential future seat utilization (seats-to-go value) is calculated. This seats-to-go value is estimated through a two-pass heuristic that attempts to optimally pack a randomly generated theoretical stream of future requests into the remaining open time-slot/table combinations. This same stream of future demand is then used to calculate the expected seats-to-go value assuming the pending request has been tentatively accepted. This process is then repeated. With each repetition, a different randomly generated stream of future requests is inputted into the heuristic in what is called Monte Carlo simulations. The final seats-to-go values for the reject and accept options are averaged over all the Monte Carlo simulations. If the expected seats-to-go for the case where the reservation is tentatively rejected cannot make up for the immediate gains of accepting the reservation, then a final decision is made to accept the request. Otherwise, the pending request is rejected.

Each pending request is received in a period $k$, where $k < T$, and thus it is known that there are $T - k$ remaining time periods before the service date, each with a preset probability that a customer will request a reservation during this slot. The number of theoretical future requests to generate for the heuristic is thus determined by the number of remaining time periods before the service day and the probability of a nontrivial request during a time period $k$.

The value of $J_k(n_k, \vec{s_k}, \vec{t_k}, s_k^*, t_k^*)$, henceforth referred to as the seats-to-go function, is the maximum expected number of additional seats booked over the remaining time periods $k$ to $T$.

The dynamic programming recursion is as follows:

$$J_T(n_T, \vec{s_T}, \vec{t_T}) = 0, \tag{2.3}$$

$$J_k(n_k, \vec{s_k}, \vec{t_k}, s_k^*, t_k^*) = \begin{cases} E[J_{k+1}(n_k, \vec{s_k}, \vec{t_k}, s_{k+1}^*, t_{k+1}^*)], \\ \text{if cannot accommodate request.} \\ \max\{s_k^* + E[J_{k+1}(n_k + 1, \vec{s_k} \cup s_k^*, \vec{t_k} \cup t_k^*, s_{k+1}^*, t_{k+1}^*)], \\ E[J_{k+1}(n_k, \vec{s_k}, \vec{t_k}, s_{k+1}^*, t_{k+1}^*)]\}, \qquad \text{otherwise } \forall \text{ k<T.} \end{cases} \tag{2.4}$$

Equation 2.3 shows the value of the seats-to-go function at the end state. At period $T$, when the day of service finally arrives, no more seats can be assigned, so the seats-to-go from that time onward is 0. The first case of Equation 2.4 corresponds to the requirement to reject the reservation request, and consider the next request, if the restaurant cannot accommodate the request. The second case of Equation 2.4 is only relevant if the request can be accommodated. It concerns the decision to accept or reject the pending reservation based on which choice returns a greater expected seats-to-go value. The expression $s_k^* + E[J_{k+1}(n_k + 1, \vec{s_k} \cup s_k^*, \vec{t_k} \cup t_k^*, s_{k+1}^*, t_{k+1}^*)]$ describes the scenario where the restaurant considers accepting the pending request. The restaurant acquires the immediate proceeds from the current request, $s_k^*$, and the expected future seat utilization, $E[J_{k+1}(n_k + 1, \vec{s_k} \cup s_k^*, \vec{t_k} \cup t_k^*, s_{k+1}^*, t_{k+1}^*)]$. This case is weighed against the reject option where it is only necessary to consider the expected future seat utilization, $E[J_{k+1}(n_k, \vec{s_k}, \vec{t_k}, s_{k+1}^*, t_{k+1}^*)]$, after rejecting the pending request. The solution to the dynamic programming problem is summarized as a threshold policy in which a request is accepted only if its immediate gains make up for the potential future decrease in the seats-to-go value,

$$s_k^* \geq E[J_{k+1}(n_k, \vec{s_k}, \vec{t_k}, s_{k+1}^*, t_{k+1}^*)] - E[J_{k+1}(n_k + 1, \vec{s_k} \cup s_k^*, \vec{t_k} \cup t_k^*, s_{k+1}^*, t_{k+1}^*)]. \tag{2.5}$$

The reservation will be accepted if there is sufficient space left in the restaurant, and the inequality in Equation 2.5 holds.

The state space grows very quickly as the number of seats in the restaurant is increased. Calculating the exact value of $J_k$ would be computationally very burdensome as it would involve simulating over all possible streams of future demand. Also, although the accuracy of the seats-to-go value increases as the number of Monte Carlo simulations increases, so too does the computational efficiency. The model's computation time is essential as a customer is not likely to wait long for a decision to be made on his or her request. Therefore, it is necessary to develop a heuristic that is fast and efficient in approximating the value of the seats-to-go function each time a new reservation request is received. The heuristic proposed in Bossert (2009), described below, takes two passes through a theoretical stream of future requests in order to quickly estimate the seats-to-go value.

### 2.2.2 Two-Pass Heuristic

On the first pass, the heuristic effectively accepts only reservation requests that are defined to be an exact fit. In order to be characterized as an exact fit, a request's party size must equal the size of the table. Further, the assignment of the given reservation to the requested time slot must not reduce the turn count of the table by more than one. Once all exact fit reservations have been placed, the heuristic moves on to the second pass which considers the requests not satisfied on the first pass. On the second pass, the restaurant accepts any request where the table size is sufficiently large (party size $\leq$ table size) and where the requested time slot is open.

### 2.2.3 Results of Accept/Reject Model

Bossert (2009) tests the performance of his proposed reservation-scheduling model through simulations at five real restaurants located in the Eastern United States. The table layouts and configurations of each restaurant were estimated. A series of 40 simulations compared the proposed model to a basic scheduling system. This basic scheduling system greedily accepts any reservation request that the restaurant can accommodate. For every simulation, each request from a randomly generated stream of reservations was passed to the dynamic program to make the corresponding accept/reject decision. A parameter was also created to allow for varying degrees of demand. The demand density parameter is the ratio of the number of pending requests to the maximum number of reservations that a restaurant can accept.

The implementation of Bossert's scheduling model produced significant increases in revenue for each of the five restaurants. The increases were as high as 11.3 percent at the highest level of demand.

# Chapter 3

# Extending the Accept/Reject Model

This chapter explains how an element of flexibility is added to both the overarching dynamic program and the heuristic that is used to estimate the seats-to-go value.

## 3.1 The Flexibility Model

The new model incorporates an element of flexibility that is absent in the accept/reject model proposed by Bossert (2009). The flexibility element allows requests to be assigned to a time slot that differs from the initially requested time. This is a realistic addition to a restaurant reservation model as there are very few restaurants, if any, that will not attempt to accommodate a request at a different time if the one that has been requested is taken. In adding this flexibility element to the model, the likelihood that a party will accept an arrival time differing from the one they have requested must be considered. The most accurate estimation of these likelihoods involves a probability distribution that is inversely related to the absolute difference between the requested and proposed arrival times, but this vector will change depending on the restaurant. A restaurant that is in very high demand will most likely have very lenient customers who are willing to take any reservation they can get. For example, three star Michelin restaurants often require reservations to be made three months in advance, and in this case, customers will accept any time they can get. Before introducing the new dynamic programming recursion, it is necessary to introduce a few new variables that help incorporate the element of flexibility. Let $\vec{P}$

be the vector containing the probabilities of customer acceptance indexed by the absolute difference in requested versus proposed arrival time. The $d^{th}$ element of $\vec{P}$, denoted $p_d$, is the probability that a customer will accept a reservation $d$ slots earlier or later than the requested time. It is important to note that $p_0$ will always be one because a customer will never reject a restaurant's offer to be seated at the time they have requested. The exact relationship relating the difference between proposed and requested times and probability of customer acceptance has yet to be determined. As a result, for the current simulations, these probabilities are estimated and sensitivity analysis has been performed.

Let the set $A_k$ compromise the set of assignments, one for each table of sufficient size, at the open time slot closest to the customer's request. Let $a$ be an arbitrary element of the set $A_k$ where each $a$ has a specific time and table associated with it. In this model, accepted reservations are assigned to a specific table and not allowed to float, so we introduce two new variables, $\vec{q_k}$ and $q^*_k$, which are defined below.

$\vec{q_k}$ **:** vector of length $n_k$, indicates the table assigned to accepted request $k$;

$q^*_k(a)$ **:** the table that is a part of the element $a \in A_k$. ($t^*(a)$ is the arrival time that is a part of element $a \in A_k$ ).

We still must store the variable $t^*_k$, which represents the requested arrival slot of the pending reservation request. It is possible that $t^*(a) = t^*_k$; this will occur when the restaurant offers the the customer their requested time slot.

The dynamics of the reject case can be seen in Equation 3.1. These dynamics mimic those of Bossert's model except that the specific table assignments contained in the vector $\vec{q_k}$ must be passed along from request to request. The dynamics of the accept case can be seen in Equation 3.2. The vector $\vec{t_k}$ must be updated with the time that the new request has been assigned. Recall that this time, $t^*(a)$, is now allowed to be different from the time the customer has initially requested. Further, the table assignment of the pending request, $q^*_k(a)$, must be added to the vector $\vec{q_k}$ of previously assigned tables.

$$(n_k, \vec{s_k}, \vec{t_k}, \vec{q_k}, s^*_k, t^*_k) \rightarrow (n_k, \vec{s_k}, \vec{t_k}, \vec{q_k}, s^*_{k+1}, t^*_{k+1}). \tag{3.1}$$

$$(n_k, \vec{s_k}, \vec{t_k}, \vec{q_k}, s^*_k, t^*_k) \rightarrow (n_k + 1, \vec{s_k} \cup s^*_k, \vec{t_k} \cup t^*(a), \vec{q_k} \cup q^*_k(a), s^*_{k+1}, t^*_{k+1}). \tag{3.2}$$

The dynamic programming recursion is thus as follows:

$$J_T(n_T, \vec{s_T}, \vec{t_T}, \vec{q_k}) = 0, \tag{3.3}$$

$$J_k(n_k, \vec{s_k}, \vec{t_k}, \vec{q_k}, s_k^*, t_k^*) = \begin{cases} E[J_{k+1}(n_k, \vec{s_k}, \vec{t_k}, \vec{q_k}, s_{k+1}^*, t_{k+1}^*)], \\ \text{if restaurant cannot accommodate request} \\ \max\Big\{ \max_{\forall a \in A_k}\{(s_k^* + E[J_{k+1}(n_k+1, \vec{s_k} \cup s_k^*, \vec{t_k} \cup t^*(a), \vec{q_k} \cup q_k^*(a), s_{k+1}^*, \\ t_{k+1}^*)])(p_{|t_k^* - t^*(a)|}) + (E[J_{k+1}(n_k, \vec{s_k}, \vec{t_k}, \vec{q_k}, s_{k+1}^*, t_{k+1}^*)](1 - p_{|t_k^* - t^*(a)|}))\}, \\ E[J_{k+1}(n_k, \vec{s_k}, \vec{t_k}, \vec{q_k}, s_{k+1}^*, t_{k+1}^*)]\Big\}, \quad \text{otherwise } \forall \text{ k<T.} \end{cases} \tag{3.4}$$

The dynamic program presented in Equations 3.3 and 3.4 differs in two respects from the dynamic program proposed in Bossert (2009). First, with the addition of flexibility, the reject option now must be weighed against the benefits of accepting the request in all the time-slot/table combinations that make up $A_k$. The dynamic recursion also must reflect the possibility that the customer could accept or reject the restaurant's offer. The first case of Equation 3.4 accounts for this possibility by scaling the seats-to-go value of the accept decision by the relevant $p_d$ value, and the reject decision by $1 - p_d$. Recall that $d$ is the difference between the requested and proposed arrival times. The criteria for accepting the request, seen in Equation 3.5, is very similar to that of Bossert's model.

$$s_k^* \geq E[J_{k+1}(n_k, \vec{s_k}, \vec{t_k}, \vec{q_k}, s_{k+1}^*, t_{k+1}^*)] - E[J_{k+1}(n_k+1, \vec{s_k} \cup s_k^*, \vec{t_k} \cup t^*(a), \vec{q_k} \cup q_k^*(a), s_{k+1}^*, \\ t_{k+1}^*)](p_{|t_k^* - t^*(a)|}) + E[J_{k+1}(n_k, \vec{s_k}, \vec{t_k}, \vec{q_k}, s_{k+1}^*, t_{k+1}^*)](1 - p_{|t_k^* - t^*(a)|})). \tag{3.5}$$

We consider only placing each request at a single time slot per table, namely the time slot that is closest to the requested time. The main reason for this limitation is computational complexity; checking every time-slot at every table would would be computationally very expensive. This approach is also an attempt to seat as many customers as possible in the time slots closest to their request, while at the same time maximizing the total people seated throughout the dining period. If placing a given request in their desired time slot is unprofitable, then the heuristic will acknowledge this with a low seats-to-go value, and the restaurant will reject the reservation. If a restaurant is in high demand, it is likely that this method will result in the restaurant rejecting many of the early requests with the hope that the later requests will be more profitable. Although when the restaurant eventually does start making offers, it is likely that most customers will be offered times close to their request.

A different model could consider the consequences of placing the pending request in every open time slot at every table. This model would intuitively seat more people and have fewer cases where the restaurant would flat out reject the customer, but at a greater computational expense.

Flexibility must also be incorporated in the approximation of the heuristic that is used to estimate the potential future seat utilization . In the next section, a heuristic is described that incorporates this added flexibility element.

## 3.2 Incorporating Time Flexibility in the Heuristic

The restaurant must decide between offering the customer a table that is an exact fit at a time differing from their request, or a table that is oversized at their requested time. In offering the former, the restaurant risks having their request rejected by the customer. In the latter case, the restaurant risks losing revenue from unused seats. In order to effectively assess the benefits of each of these two cases, the restaurant must consider the level of future demand. For a restaurant that does not anticipate much demand, the heuristic should pick the lower-risk option that guarantees revenue, the exact time fit. A restaurant that is very busy will risk rejection for an exact table fit because they know there will be other opportunities to fill the table if the customer rejects their offer.

The next section describes the heuristic that was developed to incorporate flexibility.

## 3.3 The Expected Gap Heuristic

The expected number of unused seats for a given time-slot/table assignment, we decide to refer to this value as the expected gap, is calculate using Equation 3.6:

$$E[Gap]_{ijkl} = TableSize_j - (PartySize_i * p_{|k-l|}). \tag{3.6}$$

Each request $i$ is characterized by a party size, $PartySize_i$, and some requested arrival time $k$. The restaurant looks to fit request $i$ into some table $j$ at some starting time $l$. $TableSize_j$ represents the size of table $j$, and $|k-l|$ is the absolute difference between the requested and proposed arrival times. The expected gap is minimized by simultaneously minimizing the difference between table size and party size and maximizing the value of $p_d$, the probability that the customer will accept an arrival time differing by $d$ slots from their requested time. A time-slot/table assignment that results in a perfect fitting table is profitable, but the restaurant must assess how likely the customer is to accept this assignment. In picking the time-slot/table combinations that have the lowest expected gap, the restaurant balances the risk that a customer might reject the restaurant's offer with the goal of having as few unused seats as possible.

The Expected Gap (EG) heuristic is characterized by two-passes through the theoretical stream of randomly generated future requests. The passes are executed sequentially such that a later pass cannot begin until all preceding passes have finished. The first pass of the EG heuristic only makes time-slot/table assignments where the expected gap is zero and the turn of the table is only reduced by one. Requests that were accepted by the first pass are already assigned to tables and thus the second pass does not consider them. The second pass considers each of the remaining requests one at a time. For each request, the open time-slot closest to the customer's request is found at each table. This is done to mimic the structure of the overarching dynamic program. Once this expected gap has been calculated for each time slot/table combination of a given request, the restaurant offers the customer the option with the lowest expected gap. With probability $p_d$, the customer accepts the restaurant's proposed change. If the request is accepted, the value of the seats-to-go function is updated. If the reservation is rejected, the heuristic moves on to the next request and no change is made to the seats-to-go value.

## Simple Example: Finding the Value of the Seats-to-Go Function with the EG Heuristic

Consider a hypothetical restaurant with two six-tops, table A and table B, and four dining slots. Assume a constant meal duration of two periods and an initially empty restaurant. The generated stream of future demand is summarized in Table 3.1 and the acceptance probabilities are given in Table 3.2.

The EG heuristic first employs Bossert's first pass; requests will only be accepted if they are an exact fit. Only request 2 has the party size exactly matching the table size, and we notice that placing it at either table in slots 1 and 2 will only reduce the turn count by one. Thus, only request 2 is ac-

| Request | Party Size | Arrival Time | End Time |
|---|---|---|---|
| 1 | 4 | 2 | 3 |
| 2 | 6 | 1 | 2 |
| 3 | 5 | 2 | 3 |
| 4 | 2 | 1 | 2 |

Table 3.1: The stream of future requests used in the example for the EG heuristic.

| Absolute Difference in Arrival Slot | Probability of Acceptance |
|---|---|
| 0 | 1 |
| 1 | 0.5 |
| 2 | 0.25 |

Table 3.2: The acceptance probabilities used in the example for the EG heuristic.

cepted on the first pass and without loss of generality it is placed at table A. The table statuses are updated and the result is shown in Table 3.3. The second pass considers each request one at a time and calculates the expected gap at each feasible table for the time slot closest to the customer's initial request. The first request is for slots 2 and 3 which is only open at table B. The EG for table A must be calculated assuming that request 1 is placed in slots 3 and 4. The expected gap is calculated for each assignment and can be seen in Table 3.4.

The time-slot/table combination with the lowest expected gap is offered to the customers. For Request 1, the customer will be offered their requested slots of 2 and 3 at table B. In this case, the customer is offered their requested time slot so $p_d$ is one and thus it is guaranteed that the cus-

| Slots | Table A | Table B |
|---|---|---|
| 1 | X | O |
| 2 | X | O |
| 3 | O | O |
| 4 | O | O |

Table 3.3: The table statuses after Request 1 has been satisfied.

| Request | Table | Arrival Time | Expected Gap |
|---|---|---|---|
| 1 | A | 3 | $(6-(4*0.5))=4$ |
| 1 | B | 2 | $(6-(4*1))=2$ |

Table 3.4: The expected gap for Request 1.

| Slots | Table A | Table B |
|---|---|---|
| 1 | X | O |
| 2 | X | X |
| 3 | O | X |
| 4 | O | O |

Table 3.5: The table statuses after Request 1 and 2 have been satisfied.

tomers will accept the offer. The table statuses are updated, as shown in Table 3.5, and now the heuristic moves on to Request 3.

Request 3 is for a party of five. Since Table B does not have two consecutive open time slots it no longer needs to be considered. Notice that Request 3 has requested an arrival time of time slot 2 which is open nowhere and thus we consider moving it to slots 3 and 4. The calculations for the expected gap are seen in Table 3.6. Request 3 is thus offered slots 3 and 4 at table. Since $p_1 = 0.5$, if the randomly generated number is less than 0.5, the customer will accept this reassignment. Assume that the randomly generated number is 0.42 and thus customer 3 accepts this reassignment. Table 3.7 shows the updated table statuses.

Because there are not two consecutive open time slots, it will be impossible to accept any more reservations. The value of the seats-to-go function for this example is 15.

| Request | Table | Arrival Time | Expected Gap |
|---|---|---|---|
| 3 | A | 3 | $(6-(6*0.5))=3$ |

Table 3.6: The expected gap for Request 2.

| Slots | Table A | Table B |
|---|---|---|
| 1 | X | O |
| 2 | X | X |
| 3 | X | X |
| 4 | X | O |

Table 3.7: The table statuses after Request 1, 2, and 3 have been satisfied.

# Chapter 4

# Simulations and Results

The efficacy of the Flexibility Model was tested through simulations at a small Italian restaurant in Boston. The results of these simulations are presented in this chapter.

## 4.1 Describing Input Parameters for Simulations

To test the proposed model, a set of simulations was conducted at Trattoria Toscana, a small Italian restaurant in the Boston Fenway neighborhood. Trattoria Toscana has 30 different tables, and subsets of these tables are grouped to form eight configurations. Each of the configurations has at most 10 tables, and the restaurant can accommodate at most 90 people per day.

The restaurant was assumed to have 12 dining slots, and a constant meal duration of four slots. Bossert (2009) mentions that if each slot corresponds to a half an hour, then this timing scheme accurately models restaurant operations. For the current simulations, each possible arrival time had an equal probability of being requested. In future simulations, it would be appropriate to assume some kind of peak dining time interval. It is also assumed that there is no correlation between the party size of a given request and the arrival time they ask for. The likelihood that a given party size will call in was taken from the probability distribution given in Bossert (2009). This distribution can be seen in Table 4.1. It should be noted that the largest party size that Trattoria Toscana can accommodate is eight.

The model was simulated on four different types of customers distinguished by how willing they are to accept a reservation that differs from their initial request. A "lenient" customer will accept any proposed change

| Party Size | Probability of Receiving Requests |
|---|---|
| 1 | 0 |
| 2 | 0.4 |
| 3 | 0.1 |
| 4 | 0.3 |
| 5 | 0.05 |
| 6 | 0.1 |
| 7 | 0 |
| 8 | 0.05 |

Table 4.1: The probabilities of a given party size requesting a reservation.

in reservation time and thus the acceptance probabilities for this type of customer will contain only ones. This type of customer is likely at a restaurant that is in very high demand. The model was simulated on two different "medium" customers. The first "medium" customer is very likely to accept requests that only differ by one or two time slots, but the probability of customer acceptance decreases quickly as $d$, the difference in requested and proposed arrival times, increases. The second "medium" customer is guaranteed to accept changes of only one dining slot from their requested slot. Finally, the "strict" customer only accepts offers that are exactly at their requested time-slot. This last customer is used to compare the model proposed in Bossert (2009) to the new model. The vector $\vec{P}$ of acceptance probabilities for each of the four customers can be seen in Table 4.2.

Simulations were conducted on five different streams of 60 requests. The probability that a request was generated in each of the time periods before the service date was assumed to be one for each simulation. This means that when considering a given pending request, the model knows exactly how many requests will come in the future. While this assumption may not be entirely realistic, it does not prevent us from assessing the efficacy of the model. For the five streams of 60 requests, 10 simulations was run on each of the four customer types. For each simulation, 10 Monte Carlo simulations were used within the proposed two-pass heuristic. The results were then compared to those yielded by Bossert's model and three naïve seating models. The three naïve seating models are described in the next section.

| | Probability of Customer Acceptance | | | |
|---|---|---|---|---|
| Absolute Difference in Arrival Slot | Lenient | Medium | Medium (One Slot) | Strict |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0.95 | 1 | 0 |
| 2 | 1 | 0.7 | 0 | 0 |
| 3 | 1 | 0.5 | 0 | 0 |
| 4 | 1 | 0.2 | 0 | 0 |
| 5 | 1 | 0.1 | 0 | 0 |
| 6 | 1 | 0.05 | 0 | 0 |
| 7 | 1 | 0.05 | 0 | 0 |
| 8 | 1 | 0.05 | 0 | 0 |

Table 4.2: The acceptance probabilities for Lenient, Medium, Medium (One Slot), and Strict customers.

### 4.1.1 Naïve Seating Models

To compare our model to actual restaurant operations, we examine three naïve seating models. In most restaurants it is unlikely that the hostess or maître d′ uses any sort of mathematical analysis to assess the value of a reservation request. These reservation schedulers generally have some method where they try to place the request at their desired time slot without having too many unused seats.

The first naïve model attempts to fit a given pending request at a table that will result in the fewest unused seats. If two table assignments result in the same number of unused seats, the restaurant will offer the customer the time-slot that is closest to their desired time-slot. This model simulates a situation where the maître d′ or hostess is under pressure from the restaurant to fill every table. This model is henceforth referred to as the Naïve Table Model. The second model is similar but instead emphasizes seating customers at their desired time-slot. This second naïve model attempts to fit a pending request to the time-slot that is closest to the requested time. In this case, the tiebreaker is the number of unused seats of a given table assignment. This model simulates the scenario where the maître d′ is under pressure from the customers. This model is henceforth referred to as the Naïve Time Model. The third and final naïve model employs the second pass of the EG Heuristic and is thus later referred to as the

| | Average Number of People Seated Over Ten Simulations | | | | |
|---|---|---|---|---|---|
| Pending Requests | Flexibility Model | Bossert's Model | Naïve EG | Naïve Table | Naïve Time |
| | Lenient Customer | | | | |
| Stream 1 | 84.2 | 69.3 | 62 | 67 | 57 |
| Stream 2 | 86.3 | 64.8 | 61 | 55 | 55 |
| Stream 3 | 88.8 | 66.7 | 57 | 49 | 49 |
| Stream 4 | 86.9 | 65.8 | 72 | 54 | 54 |
| Stream 5 | 86.7 | 61.3 | 65 | 55 | 52 |

Table 4.3: The results over the ten simulations for the Lenient customer.

| | Average Number of People Seated Over Ten Simulations | | | | |
|---|---|---|---|---|---|
| Pending Requests | Flexibility Model | Bossert's Model | Naïve EG | Naïve Table | Naïve Time |
| | Medium Customer | | | | |
| Stream 1 | 83.2 | 69.3 | 59.8 | 66.8 | 57 |
| Stream 2 | 82.5 | 60.5 | 61 | 54.4 | 54.6 |
| Stream 3 | 81.8 | 66.7 | 65.6 | 50.4 | 49.6 |
| Stream 4 | 84.1 | 65.8 | 67.6 | 63.7 | 62.5 |
| Stream 5 | 82 | 61.3 | 59.2 | 57.5 | 52.7 |

Table 4.4: The results over the ten simulations for the Medium customer.

Naïve EG Model. For each request, the restaurant offers the customer the time-slot/table combination that has the lowest expected gap, based on the customer's flexibility of accepting time slots other than that requested.

## 4.2 Results

The proposed model was coded in Visual Basic for Applications (VBA) with an Excel interface. The longest response time for a single request was no longer than one second, although it should be noted that Trattoria Toscana is a fairly small restaurant. The speed of the algorithm has not yet been tested on larger restaurants.

The results from the simulations can be seen in Tables 4.3–4.6. The numbers presented in these tables are the average number of people seated over 10 simulations run on five different streams of reservation requests, with

| | Average Number of People Seated Over Ten Simulations | | | | |
|---|---|---|---|---|---|
| Pending Requests | Flexibility Model | Bossert's Model | Naïve EG | Naïve Table | Naïve Time |
| | Medium (One Slot) Customer | | | | |
| Stream 1 | 83.2 | 69.3 | 56 | 62 | 57 |
| Stream 2 | 77.2 | 60.5 | 60 | 56 | 54 |
| Stream 3 | 79.8 | 66.7 | 58 | 51 | 50 |
| Stream 4 | 84 | 65.8 | 66 | 65 | 63 |
| Stream 5 | 74.8 | 61.3 | 59 | 56 | 53 |

Table 4.5: The results over the ten simulations for the Medium (One Slot) customer.

| | Average Number of People Seated Over Ten Simulations | | | | |
|---|---|---|---|---|---|
| Pending Requests | Flexibility Model | Bossert's Model | Naïve EG | Naïve Table | Naïve Time |
| | Strict Customer | | | | |
| Stream 1 | 69.8 | 69.3 | 56 | 58 | 57 |
| Stream 2 | 66.3 | 60.5 | 60 | 51 | 52 |
| Stream 3 | 69.8 | 66.7 | 59 | 51 | 50 |
| Stream 4 | 66 | 65.8 | 63 | 51 | 50 |
| Stream 5 | 56.8 | 61.3 | 61 | 57 | 53 |

Table 4.6: The results over the ten simulations for the Strict customer.

each table showing the results for a different customer type. Even though Bossert's model can only be run with strict customers, the results of these trials are presented in the results of every type of customer for means of comparison. A visual comparison of the percentage gains in seat utlization is presented in Figure 4.1.

Although a limited number of simulations were conducted, it appears that the model is successful. The Flexibility Model outperformed each of the naïve models in every simulation except one. It is clear that an element of flexibility improves the model. The percentage gains in seat utilization for the Medium and Medium (One Slot) customer simulations over the naïve models ranged from 19 percent to 30 percent and 16 percent to 27 percent respectively. Further, it appears that the addition of any element of flexibility improves the model. The percentage gains for the trials with the Lenient Customer are 19 percent to 23 percent higher than the gains made

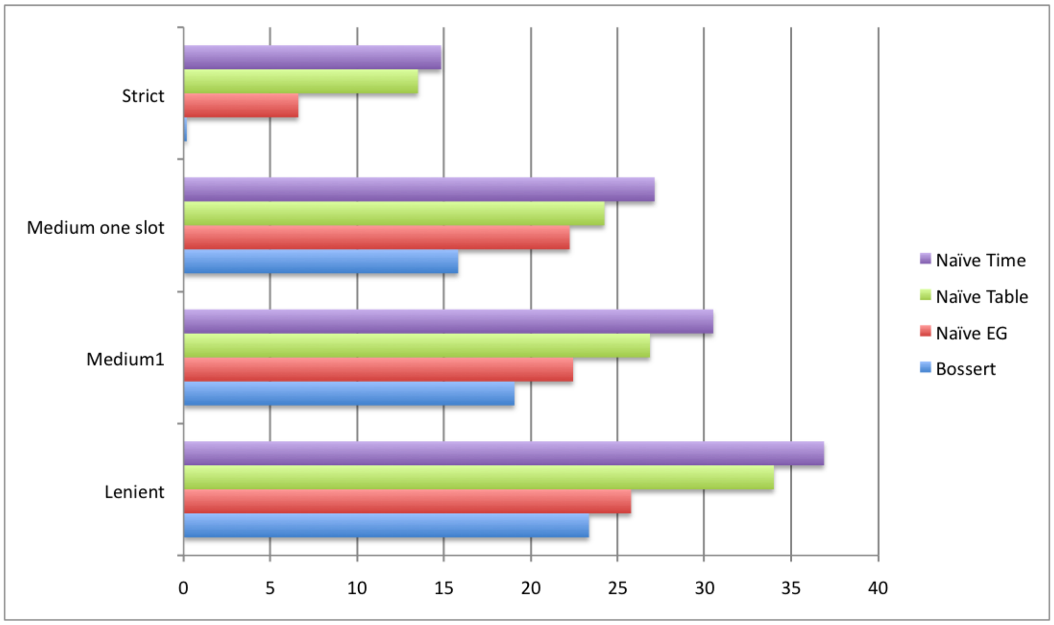

Figure 4.1: The percentage gains in seat utilization of the flexibility model compared to Bossert's model and the three naïve models.

with the Strict customer, but the differences shrink significantly when the simulations on the Lenient Customer are compared to those of the Medium (One Slot) customer. The percentage gains for in this case are only higher by 3 percent to 10 percent. Finally it is important to note the improvements of the new model over Bossert's model; these improvements range from 16 percent to 23 percent.

The reason that the flexibility element seems to be such a profitable addition is that it is now easier for the restaurant to assign requests to time slots that only reduce the turn count of a table by one. Considering that the simulations were run on a restaurant with 12 dining slots and a meal duration of four, the most profitable arrival times are 1, 5, and 9. Since the model only considers the time-slot within each table that is closest to the customer's requested time, the model relies on the heuristic to determine how an assignment to a certain time slot will affect the turn count of a given table. The results clearly indicate that the model is successful in identifying the unprofitable time slot assignment and thus it appears that the heuristic successfully communicates with the outer shell of the dynamic program.

The heuristic also was successful in identifying assignments that lead to the fewest number of unused seats, as there were few assignments made to tables that were too large.

# Chapter 5

# Future Work and Conclusions

This chapter presents the future directions of this thesis as well as a few concluding remarks.

## 5.1 Future Work

Bossert (2009) mentions that seat utilization gains of only 5 percent would likely double the profits of the restaurant. The current model produces seat utilization gains up to approximately 35 percent. It is clear that the current model needs to be explored further, as the results indicate that its use could lead to large increases in revenue for restaurants.

### 5.1.1 Short-Term Extensions

While it appears that the heuristic works well, there is still room for improvement. For the EG heuristic, the requests are processed in the order that they come, this method underestimates the number of people that can be seated. The restaurant risks seating a small party in lieu of a larger one because it is an earlier request. This problem could be remedied by sorting the requests from largest to smallest party size.

While the model was successful in the five simulations, additional simulations are necessary to prove the efficacy of the model. The current model has only been simulated on Trattoria Toscana, a small Italian restaurant. Future simulations could be used to test the model at larger restaurants with more complex configuration characteristics. Further, the current simulations only use one level of demand; 60 reservation requests. The model needs to be tested at varying degrees of demand. Finally, the current model assumes a constant meal duration across all party sizes. Not only is it plausible that this meal duration differs among party size, but it is also likely subject to some degree of randomness.

The dynamic program could also be reworked so that for each pending request, every feasible time-slot/table combination is considered. This model might seat more people, but it is unclear whether the additional computational expense is worthwhile.

### 5.1.2 Long-Term Extensions

Elements of flexibility have been incorporated with respect to dining times, but it also might be useful to consider an element of flexibility with regards to the table sizes. In very busy restaurants, it is common for the restaurant to offer to squeeze a party of four into a three-top when there are no larger tables available. Similar to the acceptance probabilities, another vector could be developed detailing the probability of customer acceptance when the restaurant offers to squeeze customers into tables that are slightly too small.

Finally, overbooking strategies and walk-in customers could also be incorporated into the model.

## 5.2 Conclusions

The proposed Flexibility Model appears to be a successful approach to optimizing restaurant-reservation scheduling. While Bossert's reservation scheduling model showed significant improvements over naïve seating models, the fact that it fails to incorporate a customer's flexibility in accepting time slots other than those requested is a significant shortcoming. Not only is this element of flexibility important for an accurate representation of the restaurant-reservation scheduling process, but its addition also allows the restaurant to seat more people throughout the dining service. It is clear that the flexibility element is an essential addition to the model based on the recorded seat utilization gains ranging from 15 to 35 percent.

# Bibliography

John M. Bossert. Yield Management of Configurable Restaurants. *unpublished manuscript*, 2009.

Sheryl E. Kimes, Richard B. Chase, Summee Choi, Philip Y. Lee, and Elizabeth N. Ngonzi. Restaurant Revenue Management: Applying Yield Management to the Restaurant Industry. *Cornell Hotel and Restaurant Administration Quarterly*, 39:32–39, June 1998.

Sheryl E. Kimes, Deborrah I. Barrash, and John E. Alexander. Developing a Restaurant Revenue-Management Strategy. *Cornell Hotel and Restaurant Administration Quarterly*, 40:18–29, October 1999.

Jeffrey I. McGill and Garrett J. Van Ryzin. Revenue Management: Research Overview and Prospects. *Transportation Science*, 33:233–256, May 1999.

Serguei Netessine and Robert Shumsky. Introduction to the Theory and Practice of Yield Management. *INFORMS Transactions on Education*, 3(1): 335–347, January 1999.

Romy Shioda. *Restaurant Revenue Management*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA 02139, May 2002.

Barry C. Smith, John F. Leimkuhler, and Ross M. Darrow. Yield Management at American Airlines. *Interfaces*, 22:8–31, February 1992.

Gary M. Thompson. Optimizing a Restaurant's Seating Capacity: Use Dedicated or Combinable Tables? *Cornell Hotel and Restaurant Administration Quarterly*, 43(3):48–57, August 2002.

Gary M. Thompson and Sheryl E. Kimes. An Evaluation of Heuristic Methods for Determining the Best Table Mix in Full-Service Restaurants. *Journal of Operations Management*, 23(2):599–617, 2005.

Gary M. Thompson and Robert J. Kwortnik, Jr. Pooling Restaurant Reservations to Increase Service Efficiency. *Journal of Service Research*, 10(4):335–346, March 2008.

Alfio Vidotto, Kenneth N. Brown, and J.C. Beck. Managing Restaurant Tables Using Constraints. *Knowledge-Based Systems*, 20:160–169, 2007.