An Evaluation of Heuristic Methods for Determining the Best Table Mix In Full-Service Restaurants.

Sheryl E. Kimes
School of Hotel Administration
Cornell University
Ithaca, NY
sek6@cornell.edu

Gary M. Thompson
School of Hotel Administration
Cornell University
Ithaca, NY
gmt1@cornell.edu

Abstract

Little research has been done on the optimal mix of supply in service businesses that maximizes revenue. Our research context is the full-service restaurant table mix problem. This problem, which is quite new to the literature, finds the optimal number of different size tables for a restaurant to maximize its value (revenue or contribution) generating potential. Specifically, we examine the effectiveness of eight heuristic techniques for the problem using two experiments. The first experiment uses data from a 240-seat full-service restaurant to evaluate all eight heuristics, while the second experiment investigates the performance of selected heuristics under a broader set of environmental factors. The results of our first experiment showed that the better of the simulated annealing heuristic variants yielded the optimal solution in seven of eight test problems, averaging within 0.1% of optimal. Our second experiment showed that the simplest of the models we investigated yielded solutions within 1% of the simulated annealing solution. Finally, we observed that altering the table mix on a daily basis increased performance by over 1% compared to maintaining the optimal weekly table mix.

Keywords: yield management, capacity management, service operations, simulation

1. Introduction

Revenue management (RM) has been defined as selling the right inventory unit to the right customer at the right time and at the right price (Smith et al., 1992). It typically focuses on demand management through manipulating price and inventory controls, but rarely considers the impact of managing supply or developing the best mix of inventory units. The best RM practices may still lead to sub-optimal results if supply is not well configured.

As examples, consider hotels and airlines. Hotels use rate and length of stay controls, but do not generally consider whether they have the best mix of room types. Airlines use fare and origin–destination controls and implement fleet assignment algorithms (Rexing et al., 2000), but do not usually consider whether they have the best mix of plane sizes or whether the planes are optimally configured. Restaurants have only recently begun to implement RM and have concentrated on managing customer dining time (Kimes et al., 2002), using demand-based pricing (Kimes and Wirtz, 2002) and increasing kitchen productivity (Sill, 1991), but with the exception of Kimes (2004) and Kimes and Thompson (2004), have not really considered how their table mix affects revenue.

Little research has been done on the profit- or revenue-maximizing mix of supply in any industry. While the revenue management research has concentrated on the most profitable way in which to allocate demand to available supply, and the capacity management literature has focused on matching supply and demand in the most efficient way, we have found very little research that addresses the mix of supply that will help maximize revenue for a given demand.

In this paper, we concentrated on the optimal table mix in the restaurant industry. If a restaurant can use its demand data to develop a table mix that can help it serve more customers during busy periods, it can increase revenue. The revenue potential for an improved table mix for the restaurant industry is quite large. In the US, $ 147 billion was spent in full-service restaurants (those having a wait staff) and US$ 115 billion in limited service (fast-food) restaurants (National Restaurant Association, 2002).

2. Literature review

Two streams of literature seem relevant to the supply mix problem: revenue management and capacity management. Revenue management is a specialized form of capacity management that seeks to manage demand to maximize revenue or contribution, while capacity management attempts to better manage supply and demand for improved efficiency.

2.1. Revenue management

Revenue management (RM) was originally developed in the airline industry (Smith et al., 1992) and has gained widespread acceptance in the hotel industry (Baker et al., 2002 and Hanks et al., 1992), the cruise industry (Lieberman and Dieck, 2002) and rental car industry (Carroll and Grimes, 1995 and Geraghty and Johnson, 1997). Companies using RM have reported revenue increases of 2–5% (Hanks et al., 1992 and Smith et al., 1992). RM efforts generally revolve around inventory allocation and overbooking approaches (Baker et al., 2002, Belobaba, 1989, McGill and van Ryzin, 1999 and Weatherford and Bodily, 1992) and pricing initiatives (Bitran and Mondschein, 1997 and Gallego and van Ryzin, 1994).

The expected marginal seat revenue (EMSR) and bid price models are the most widely used approaches and have been the focus of much research (McGill and Van Ryzin, 1999). Regardless of the allocation method used, detailed forecasts of customer arrival, cancellation and no-show behavior are required (i.e. day of week, time of day and flight for the airline industry or day of week, length of stay and room type in the hotel industry).

The literature on restaurant revenue management is fairly recent (Kimes et al., 1998 and Kimes et al., 1999) and almost all articles have focused on arrival (Bertsimas and Shioda, 2003, Kimes, 2004 and Kimes and Thompson, 2004) and duration (Kimes et al., 1999 and Kimes et al., 2002) control. Limited research has been conducted on the optimal supply mix (Kimes and Thompson, 2004, Thompson, 2002 and Thompson, 2003). Pricing, although an important aspect of any revenue management strategy, has received limited attention (Kimes and Wirtz, 2002 and Kimes and Wirtz, 2003). Since the focus of our research is on the optimal table mix, an important element of any arrival control strategy, the articles on arrival control and supply mix are the most relevant.

Bertsimas and Shioda (2003) studied optimal table assignment rules for a small, fictional restaurant. Table assignment rules considered included first-come-first-served, full nesting, one-up nesting and no-nesting. With the first-come-first-served assignment rule, parties were seated on a first-come-first-served basis at the next available table that was capable of fitting the party (i.e. a party of four could not be seated at a table for two). The full nesting rule allowed parties of size k to be seated at a size k' table for $k' \geq k$. If there were several possible table sizes available, the party was seated at the smallest table size. The one-up nesting rule seated parties of size k at either a table of size k or at the next largest table size if size k tables were unavailable. The no-

nesting rule only seated parties at the right table size (i.e. parties of 4 could only be seated at tables for 4).

They developed and tested three different models (integer programming, stochastic integer programming and approximate dynamic programming) for each of the nesting rules and found that the approximate dynamic programming model achieved higher revenues without an increase in waiting time. While their research is interesting, they assumed that the table mix was static.

Thompson (2002) addressed the issue of combinable tables. Specifically, he studied whether it is more profitable to have tables dedicated to serving parties of a particular size (i.e. parties of 1 or 2 can be served at 2-tops, parties of 1–4 can be served at 4-tops) or whether the restaurant should use tables that can be combined as needed based on party size. He used simulation to study the impact of using five levels of combinability (0%, 10%, 30%, 50% and 100%) in two different restaurant sizes (50 and 200 seats) for three different mean party sizes (2.5, 3.5 and 4.5 people).

He found that combinable tables worked better only for smaller restaurants with small mean party sizes and that dedicated tables worked best for all other restaurants. This outcome can be explained by the loss of productivity that occurs when some tables are held out of service until adjacent tables become available (so that the tables can be combined to seat a large party). In later research (2003), he studied the optimal configuration of combinable tables and found that better configurations used longer sequences of smaller combinable tables and that effective configurations could result in a 1.4% improvement in revenue.

Kimes and Thompson (2004) used data from a 230-seat Chevys Freshmex Restaurant in conjunction with a simulation model to test all possible combinations of table sizes. They found that an optimal table mix allowed the restaurant to serve up to 35% more customers without increasing waiting time as compared to the restaurant's existing table mix. By implementing an improved table mix, the restaurant was able to increase revenue by 5.1% with a payback period of less than one year. A full discussion of the data and the implementation process is included in Kimes (2004).

2.2. Capacity management

Research in capacity management has primarily addressed methods of managing demand to better match supply and ways of manipulating supply to better manage demand, but has rarely considered the impact of redesigning supply (Klassen and Rohleder, 2001, Lovelock, 1992, Ng et al., 1999 and Sasser, 1976). In an extensive search of the literature we found few mentions of the optimal supply mix or the impact that supply mix has on revenue: the optimal amount of supply was discussed in a quickservice restaurant environment (Banker and Morey, 1993), the rental car industry considers both the optimal amount and configuration of supply (Carroll and Grimes, 1995 and Geraghty and Johnson, 1997), and the labor scheduling literature has some models addressing the issue (Goodale et al., 2003 and Thompson, 1995).

The design of supply has four components: (1) the number of facilities; (2) the size of those facilities; (3) the supply mix; and (4) the flexibility of that supply mix (Kimes and Thompson, 2004). Based on the research of Bertsimas and Shioda (2003), we believe there is fifth element: the supply assignment rule. Restaurants must decide which parties to seat at which tables, self-storage companies must decide which customers to place in certain spaces and rental

car companies must decide which clients should receive certain-sized cars. While Bertsimas and Shioda (2003) propose four different table assignment rules (first-come-first-served, fully nested, one-up nesting and no nesting), there are several other rules including, for example, assigning a table to the largest waiting party that fits. As discussed in Bertsimas and Shioda (2003), the table assignment rule used can have a significant revenue impact.

The optimal supply mix can have considerable revenue impact. For example, in the airline industry, the optimal mix of plane sizes can help an airline operate more efficiently and increase profitability, or in the hotel industry, the optimal mix of room types can greatly affect the financial results achieved. While the revenue management research has concentrated on the most profitable way in which to allocate demand to available supply, we have found very little research (with perhaps the exception of the flight assignment program (Rexing et al., 2000)) that addresses the optimal mix of supply. Other industries face similar supply mix problems. For example, self-storage facilities must decide on the optimal mix of different size storage spaces, and performing arts centers and stadiums must decide on the dynamic optimal partition of seats to different market segments.

In the restaurant context, the optimal mix of supply is the number of each size table that should be used. Specifically, what is the best mix of 2-tops (tables for 2), 4-tops (tables for 4), 6-tops (tables for 6) and 8-tops (tables for 8)? The ideal table mix is affected by a number of factors including space constraints, party characteristics, restaurant layout, and table combinability. Interestingly, from a space perspective, tables commonly require space proportional to their number of seats, the number of tables used for a given seat capacity is immaterial since a 2-top typically takes fifteen square-feet (including space for the table, chairs

and circulation), a 4-top takes about 30-square-feet and a 6-top takes about 45-square-feet (Robson, 2004). Seat capacity is typically governed by facility size and corporate strategy, while party size is most probably a result of local area demographics, restaurant type and marketing. Party size distribution is stochastic and can vary by day of week and time of day. Dining duration and expenditure patterns are largely dictated by the type of restaurant (i.e. fine dining, fast-food), by internal service delivery systems and by type of customer (i.e. business or pleasure) and can exhibit a high variance across customers. Combinable tables are separate, adjacent tables that can be combined to seat larger parties.

Restaurants face the table mix problem both before and after construction. In the former case, they must use the predicted customer mix to address the table mix problem; while in the latter case, they can periodically re-evaluate their table mix based on the actual customer mix. Re-evaluating a restaurant's table mix is possible because the cost of changing the table mix is not prohibitively high if all 2-tops are used, since those tables can be reconfigured into the desired layout during slow times and since, in general, no additional labor cost is required and if the restaurant is configured with all 2-tops, it can reconfigure those tables into desired layout). In our earlier research, we used complete enumeration to develop an optimal table mix for an existing Chevys restaurant in the Southwestern US (Kimes and Thompson, 2004). Management implemented a table mix that differed slightly from our recommended mix and that mix led to a 5.1% increase in overall revenue. Although complete enumeration yielded optimal results, it was computationally time consuming. This spurred our interest in finding fast, effective heuristic methods for solving the table mix problem.

3. Solution approaches

After considering the merits of various approaches to table mix optimization, we settled on simulation as a tool to evaluate table mixes, since it is ideally suited to incorporate the stochastic nature of a restaurant. We developed a simulation model, which we call TABLEMIX (Thompson, 2002 and Thompson, 2003) that simulates the use of restaurant tables over a specified dining period.

In addition to complete enumeration of the table mix alternatives, we formulated three integer-programming based models: a naïve model, a time-based model, and a revenue-management-based model. We also developed two variants of a simulated annealing heuristic for their applicability to this problem. We describe each of the approaches below. We will first define our data.

3.1. Data definitions

Below we define indices, sets, constants and variables which we use in describing our table mix heuristics.

Indices are:

d: index for days

i, j: index for table sizes

p: index for party sizes

t, t': index for time periods

Sets are:

$CoversTime_{pt}$: time periods in which a party of size p can commence dining and still require a table in time period t, customer arrival patterns varied, and depending on the dining duration (Dur_p) and time period (t) used durations, customers might be present for multiple time periods. For example if dining duration was 1.2 h, and the time periods were 1 h, the customers would be present for one full time period plus 20% of the next. For computational simplicity, all customers were assumed to arrive at the beginning of a time period.

Days: days (or meal periods) under consideration

Parties: party sizes

$PartyFits_i$: party sizes that will fit in table size i. For example, a party of size 3 or 4 would fit at tables of 4, 6 or 8, but would not fit at a table for 2.

$TableFits_p$: table sizes that will hold a party of size p. For example, a party of four could be seated at 4-top and larger tables, but not a 3-top or smaller table.

Tables: allowed table sizes (measured as a seat count)

Times: daily time periods

Constants are:

b_p: probability of an arriving party being of size p

Dur_p: mean dining duration (in minutes) for parties of size p

ExpParties$_{pd}$: expected number of arriving parties of size *p* during the peak demand period on day *d*

ExpParties$_{pdt}$: expected number of arriving parties of size *p* during the time period *t* on day *d*

$$\text{NDBSeatProportions}_i = i \sum_{p \in \text{PartyFits}_i} b_p$$

Seats: total seats in the restaurant

NDBSeatProportionsi is an estimate, that does not consider dining duration, of the demands placed on tables having i seats.

$$\text{NDBIdealSeats}_i = \text{Seats} \frac{\text{NDBSeatProportions}_i}{\sum_{j \in \text{Tables}} \text{NDBSeatProportions}_j}$$

NDBIdealSeatsi: is an estimate, that does not consider dining duration, of the ideal total number of seats in tables having i seats.

$$\text{DBSeatProportions}_i = i \sum_{p \in \text{PartyFits}_i} (b_p \text{Dur}_p)$$

DBSeatProportionsi: is an estimate, that consider dining durations, of the demands placed on tables having i seats.

$$\text{DBIdealSeats}_i = \text{Seats} \frac{\text{DBSeatProportions}_i}{\sum_{j \in \text{Tables}} \text{DBProportions}_j}$$

$DBIdealSeats_i$: is an estimate, that considers dining duration, of the ideal total number of seats in tables having *i* seats.

$PeakMins_d$: length of the peak demand period on day *d*, in minutes

$Value_p$: mean value (i.e., revenue or contribution) for parties of size *p*

$Value_{pd}$: mean value (i.e., revenue or contribution) for parties of size *p* on day *d*

Variables are:

x_i: number of tables with *i* seats

s_i^- si−: seat shortage of tables with *i* seats

s_i^+ si+: seat surplus of tables with *i* seats

w_{pid}wpid: number of parties of size *p* seated at tables with *i* seats on day *d*

w_{pidt}wpidt: number of parties of size *p* seated at tables with *i* seats in time period *t* of day *d*

3.2. Complete enumeration (Enum)

We used the TABLEMIX enumeration functionality to evaluate all unique table mixes of a restaurant. The large number of table mixes becomes quite large because all combinations of 2-tops, 4-tops, 6-tops and 8-tops must be considered. For example, a restaurant of 120 seats would have 1815 table mix combinations using all 120 seats. Larger restaurants have even more combinations: doubling the size of the restaurant to 240 seats increases the number of table-mix

combinations by almost 650%, to 13,561. Enumerating all possible table mixes enabled us to identify the optimal solution for each problem and also to easily assess the solution quality of the different heuristics we tested.

3.3. Naïve integer programming models (NaïveIPs)

We used two versions of a Naïve integer programming model. NaïveIP-A is based on calculations presented in Thompson (2002) and Kimes and Thompson (2004) and only includes information on party size mix, total number of seats, and allowable table sizes. NaïveIP-B is similar to NaïveIP-A, but accounts for dining durations that vary by party size. We have also converted the calculations to an IP model since Thompson (2002) did not offer suggestions on how to round fractional solutions to integral numbers of tables. NaïveIP-A is:

$$\min \sum_{i \in \text{Tables}} (s_i^- + s_i^+)$$

Subject to:

$$ix_i + s_i^- - s_i^+ = \text{NDBIdealSeats}_i \quad \text{for } i \in \text{Tables}$$
$$\sum_{i \in \text{Tables}} ix_i \leq \text{Seats}$$
$$x_i \geq 0 \text{ and integer} \quad \text{for } i \in \text{Tables}$$

The objective for NaïveIP-B is the same as for NaïveIP-A, but the first constraint set is different:

Min (5)

Subject to:

$$ix_i + s_i^- - s_i^+ = \text{DBIdealSeats}_i \quad \text{for } i \in \text{Tables (7) and (8)}$$

The decision variables in NaïveIP-A and NaïveIP-B are the number of each size table to use (the x_i) and the deviation variables (the $s{-}i$ and $s{+}i$), which are used to measure differences between the actual and ideal number of seats allocated to each size table. The objective of both models (5) minimizes deviations between the actual and ideal total number of seats allocated to each size table. Basically, the objective has the goal of balancing capacity and demand, without regard to the value of different size parties.

NaïveIP-A has three constraint sets. Constraint set (6) includes information on party probabilities and measures the difference between the actual number of seats allocated to each table size and the non-duration-based ideal number of seats for that table size (2). Constraint set (7) ensures that the sum of the number of seats allocated to each table size does not exceed the number of seats in the restaurant. Constraint set (8) ensures the integrality and non-negativity of the table variables.

Similarly, NaïveIP-B has three constraint sets, two of which (7) and (8) are common to NaïveIP-A. Constraint set (9) differs from constraint set (6), only in that (9) uses duration-based estimates of the ideal number of seats for each table size.

Key inputs to NaïveIP-A and NaïveIP-B are the non-duration-based and duration-based ideal number of seats to be allocated to each table size (the NDBIdealSeats_i and DBIdealSeats_i, respectively), which are based on the goal of matching the table mix to customer demand. These calculations (2) and (4), are based on the proportion of total seats in the restaurant that each particular table size should receive. To use NaïveIP-A and NaïveIP-B for the development of the

weekly solution, one simply averages across the days the probability by party size (the b_p) and, for NaïveIP-B, the durations by party size (the Dur_p).

NaïveIP-A and NaïveIP-B use the most aggregated data of the three IP models. This is an advantage, in that it results in the models requiring few decision variables and consequently being easy to solve. However, the high level of aggregation may be a disadvantage, in that detailed information about which parties are seated at which tables is missing and the revenue associated with each party is not used, and so the models may yield inferior solutions. In addition, performing a sensitivity analysis on NaïveIP-A and NaïveIP-B provides less useful information than do the two other IP models. In earlier research (Kimes and Thompson, 2004), we found that the NaïveIP-A model produced the optimal solution, but believed that the model was not robust in most circumstances.

3.4. Time-based integer programming model (TimeIP)

Compared to the NaïveIP models, TimeIP has a lower level of data aggregation. TimeIP is:

$$\max \sum_{d \in \text{Days}} \sum_{p \in \text{Parties}} \sum_{i \in \text{TableFits}_p} \text{Value}_{pd} w_{pid}$$

Subject to:

$$\sum_{p \in \text{PartyFits}_i} \text{Dur}_p w_{pid} \leq \text{PeakMins}_d x_i \quad \text{for } d \in \text{days},\ i \in \text{Tables}$$

$$\sum_{i \in \text{TableFits}_p} w_{pid} \leq \text{ExpParties}_{pd} \quad \text{for } d \in \text{days},\ p \in \text{Parties}$$

$$w_{pid} \geq 0 \text{ and integer} \quad \text{for } p \in \text{Parties},\ i \in \text{TableFits}_p,\ d \in \text{days} \quad \text{and (7) and (8)}$$

TimeIPs decision variables are the number of each size table (the x_i) and the number of parties of each size that are served at each size table each day (the *wpid*). TimeIPs objective (10) is to maximize the value of parties served. Constraint set (11) limits the amount of time that each table size can be used, per day. Constraint set (12) limits the number of parties served of each size on each day to the number of parties of that size expected to arrive on the particular day. Constraint set (13) imposes the non-negativity and integrality restrictions on the party-to-table allocation variables. Constraint set (7) applies the seat capacity for restaurant, while constraint set (8) ensures the non-negativity and integrality of the table quantity variables. By controlling the set *days*, the days to be considered, TimeIP can be easily modified for determining the table mix for a single day or for a week.

TimeIP aggregates data less than the NaïveIP models, which should result in it yielding superior solutions compared to the NaïveIP models. The structure of TimeIP is such that its LP relaxation can yield sensitivity analysis information about the value to the restaurant of additional seats or time. TimeIP requires more integer variables than does the NaïveIP models, so we expected it to be slower to solve. Also, by aggregating customer arrival data into a single daily planning interval, TimeIP fails to represent the stochastic nature of party arrivals, party size, and service durations. In addition, its lack of revenue data may lead to suboptimal results.

3.5. Revenue management integer programming model (*RevMgtIP*)

For our next IP-based model, we adapted the linear programming model commonly used for hotel revenue management (Baker et al., 2002 and Choi and Kimes, 2002). In the hotel industry, the decision variables are the number of customer arrivals to accept per day at a particular rate category (RC) and a particular length of stay (LOS). The linear programming

model has two types of constraints: capacity (one constraint for each day) and demand (one constraint for each RC and LOS combination). RevMgtIP follows a similar format, but the LOS is the dining duration, the capacity is the number of seats available, and the rate is the average check (or some other measure of party value). RevMgtIP is:

$$\max \sum_{p\in \text{Parties}} \sum_{i\in \text{Tables}} \sum_{d\in \text{Days}} \sum_{t\in \text{Times}} \text{Value}_{pd} w_{pidt}$$

Subject to:

$$\sum_{p\in \text{PartyFits}_i} \sum_{t'\in \text{CoversTime}_{pt}} w_{pidt'} \leq x_i \quad \text{for } i \in \text{Tables},\ d \in \text{days},\ t \in \text{Times}$$

$$\sum_{i\in \text{TableFits}_p} w_{pidt} \leq \text{ExpParties}_{pdt} \quad \text{for } p \in \text{Parties},\ d \in \text{days},\ t \in \text{Times}$$

$$w_{pidt} \geq 0 \text{ and integer} \quad \text{for } p \in \text{Parties},\ i \in \text{Table Fits}_p,\ d \in \text{days},\ t \in \text{Times} \quad \text{and (7) and (8)}$$

RevMgtIPs decision variables are the number of each size table (the x_i) and the number of parties of each size that are served at each size table during each daily time period (the *wpidt*). Its objective (14) is to maximize value (whether it be revenue or contribution). Constraint set (15) ensures that, for each day and time period, the total number of occupied tables of each size does not exceed the model's choice of the number of that size table. The constraint recognizes that parties staying over from previous periods can continue to use tables. Constraint (15) is slightly different in form compared to the capacity constraints in previous RM models. The difference is that in earlier RM models, capacity is a given, so the right side of constraint (15) is a constant (for example, equal to the available number of a certain type room in a hotel). In the right side of constraint (15), however, capacity is a variable, which the model itself determines.

Constraint set (16) ensures that the number of customers of each party size for whom service commences in each daily time period does not exceed the number of parties of that size expected during that period. Constraint set (17) ensures an integer, non-negative solution of the party allocation variables. Constraint set (7) applies the seat capacity for restaurant, while constraint set (8) ensures the non-negativity and integrality of the table quantity variables. RevMgtIP can either be run for all days of week or for any combination of days desired by controlling the set *Days*.

The LP relaxation of RevMgtIP can lead to interesting sensitivity analysis results. As with the hotel bid-price method (Baker and Collier, 1999), the shadow prices of the overall capacity constraint (constraint (7)) and table capacity constraints (constraint (15)) respectively can be used to assess the value of a seat and for determining the value of a table during different time periods. For example, with the hotel bid price model, the minimum rate available for a certain length of stay can be calculated by averaging the shadow prices for the capacity constraints for those days. Although most restaurants do not use variable pricing to the extent that airlines and hotels do, they could use the shadow prices for the time period capacity constraints to determine the minimum price (or maximum discount allowable) which should be available. Bertsimas and Shioda (2003) also discussed the application of the hotel bid-price model to the table assignment rule and solved the LP relaxation of the IP model and made seating decisions based on the difference between the expected party revenue and the sum of the dual prices corresponding to the times at which the party would be using the table.

We investigated three variants of RevMgtIP, which differed in the length of daily time periods. RevMgtIP15 used 15 min time periods, RevMgtIP5 used 5 min time periods and

RevMgtIP3 used 3 min time periods. Smaller time intervals allow for a closer approximation of the true processing of customers. This led us to expect that the model would perform better with short periods. However, as the number of periods increases, the model requires far more variables, making it more difficult to solve. Also, party demand becomes more granular, which may result in poor performance.

RevMgtIP had the most detailed data of the three IP models, which led us to believe that it would perform better than the NaïveIP models and TimeIP. However, it also has four potential limitations. First, its increased number of integer variables could increase solution time. Second, it treats dining duration as fixed by party size. Third, it assumes that party arrivals are deterministic and that they occur at the beginning of each time interval. Fourth, RevMgtIP does not allow parties to wait for service: if service does not commence for a party in the period in which it arrives, the party is lost. This may become problematic with short time intervals, when by necessity the party demand is more granular.

3.6. Simulated annealing heuristic (SimAnneal)

Our simulated annealing heuristic, SimAnneal, incorporates a simulated annealing front-end on the TABLEMIX restaurant simulator as a back-end solution evaluator. The pseudo code of the heuristic is presented in Table 1. Some items of note in the pseudo code are as follows. First, we decremented the temperature parameter every two iterations, which resulted in a total of 49 decrements given our 100 iteration limit. Second, because evaluating a table mix is computationally intensive, we kept a list of mixes that we had already evaluated and ensured that we never evaluated the same mix twice. On this problem, at least, it was much quicker to check the list than to simulate the use of the mix. Finally, we did not do any particular tuning of the

parameters (e.g. *temp*, *cooling*, *DropProp*, probabilities of selecting different table sizes). However, our choices for the general annealing parameters (*temp*, *cooling*, *DropProp*) were those that have served us well in a variety of other simulated annealing heuristics (Thompson, 1996 and Brusco et al., 1997).

We implemented two versions of SimAnneal, which differed solely in their initial table mix. SimAnneal-S (for scratch) started with no tables and used the logic for adding tables previously discussed to arrive at its initial table mix. Conversely, SimAnneal-N started with the better of the table mixes identified by the NaïveIP models.

SimAnneal requires access to a restaurant simulator. The disadvantage of this is that one must build, or otherwise obtain, a simulator. The advantage of doing so, however, is that it enables SimAnneal to incorporate the stochastic nature of party arrivals, party sizes and service durations. This, combined with the fact that simulated annealing has been successfully applied in a wide variety of optimization problems, made us expect that SimAnneal would yield good solutions.

4. Experiment 1

4.1. Problem setting

In order to study the table mix problem, for our first experiment we obtained detailed data from a busy 240-seat restaurant located in a shopping mall in California. This restaurant had 2 two-tops, 56 four-tops, and 2 six-tops. This table mix was similar to other restaurants in the chain and had been selected by a restaurant designer for aesthetic and efficiency reasons. During the busy dinner period (6–8 p.m.), customers typically waited for a table for 30–60 min. The

majority of parties contained one or two people and, across the week, the average dining time was 49.5 min, with a US$ 36.36 average check per party, or US$ 13.88 per person.

Not surprisingly, demand varied by day of week and time of day (Fig. 1). The party mix also varied by day of week (Fig. 2), and dining duration and average check varied by party size (Table 2 and Table 3).

We calculated seat occupancy by day of week and time of day (Fig. 3). Hourly seat occupancy was calculated by dividing the hourly revenue per available seat hour (RevPASH) by the average check per person for that hour and then multiplying by the average meal duration (in hours) for that hour.

4.2. Problem scenarios

We used the daily data from the restaurant to create a set of eight problem scenarios in Experiment 1. Seven of the scenarios corresponded to situations in which the restaurant had a single peak demand day that would determine the optimal table mix. Because of the differences in demand data across days of the week at the study restaurant, we thought that the optimal table mix might also vary by day of week. The eighth scenario was one in which all days exhibited peak demand periods and the optimal table mix was based on what worked well across the entire week. For each scenario, we simulated 150 weeks of operation.

Optimizing a restaurant's table mix only matters if a restaurant is capacity constrained. To ensure that the restaurant was capacity constrained, we set the total minutes of customer demand equal to a 100% seat utilization during the daily peak 2 h periods. We also had a 90 min ramp-up to the 2 h peak period. Fig. 4 shows the demand levels we used, measured in the number of

parties arriving per 15 min interval, by day. The party arrival rates shown vary by day because the mean party size and mean dining duration differs by day, but seat utilization was held constant. The party arrival rates shown in Fig. 4 differ from those shown in Fig. 1, since our objective in creating the arrival rates shown in Fig. 4 was to ensure that the restaurant was capacity constrained.

4.3. Results

We used the TABLEMIX simulator to evaluate all 13,561 table combinations. The large number of table mixes occurred because we considered the use of 2-, 4-, 6- and 8-tops and because the restaurant had a total capacity of 240 seats. Revenue ranged from a low of US$ 20,624 with 30 eight-tops to a high of US$ 44,722 with the optimal mix of 56 two-tops, 24 four-tops, 4 six-tops and 1 eight-top. There were 105 mixes within 1% of optimal and 292 mixes within 2% of optimal.

4.3.1. Solution times

The techniques varied in the time required to develop solutions (Table 4). The times to determine the table mix for a single day ranged from about one second, for the NaïveIP models, to 139 min, for Enum. RevMgtIP and TimeIP were also very fast. Since SimAnneal was evaluating 100 different table mixes, the time it required was about 100/13,561, or 0.74% of the time required by Enum. Enum, RevMgtIP, TimeIP and SimAnneal all required more time to develop table mix solutions for the weekly problem compared to single-day solutions. The jump in solution time for RevMgtIP3 was particularly pronounced: for the weekly problem it required more time than did SimAnneal. NaïveIP required the same amount of time for both approaches

since it did not require any more decision variables in its weekly form (recall that all the NaïveIP models require is averaged weekly data inputs).

4.3.2. Revenue and table mix

The recommended table mixes from each method are reported in Table 5. The resulting table mixes varied among solution methods, but all had between 48 and 56 two-tops, between 22 and 25 four-tops, between 4 and 6 six-tops, and between 0 and 2 eight-tops.

Table 6 reports the revenue from the recommended table mixes for each day of week and for the entire week. Table 7 expresses the revenue as proportions of optimal. Optimal solutions to the weekly problem were found by both simulated annealing approaches. The other solution methods all provided an answer within 2% of optimal. NaïveIP-A had the best performance followed by TimeIP, RevMgtIP5, RevMgtIP3 and NaïveIP-B, and RevMgt15.

Simulated annealing also had the best performance on the daily problem. SimAnneal-N, which uses the NaïveIP solution as a starting point, found the optimal solution for six out of the seven single days. Its worst performance on any day was Thursday, when the solution it found was 0.13% below optimal. When allowed to run for more iterations, SimAnneal-N found the optimal on Thursday on its 112th iteration. SimAnneal-S was the next best performer, followed by TimeIP, NaïveIP-A, RevMgtIP5, RevMgtIP3, RevMgtIP15, and NaïveIP-B.

When a separate table mix was calculated for each day of the week, the resulting revenue obtained from all but the Naïve-B approach was between 1.0% and 1.7% higher than for a single mix used for the entire week. This suggests that a busy restaurant with highly variable demand should consider different table configurations for each day of week and possibly meal period.

The revenue values reported in Table 6 that are achieved by Enum translate to approximately 80% of the possible revenue that could have been achieved, had all arriving parties been served (which, recall, was set based on a 100% seat utilization). Thus, the 80% revenue figure translates to an 80% seat utilization value, which is comparable to the typical maximum seat utilizations that have been observed in earlier research (Thompson, 2002 and Thompson, 2003). Given our experimental assumptions, it also translates into an 80% customer service level.

4.4. Discussion

While the optimal result can be found from enumeration, the sheer time involved renders this solution method impractical for most restaurants or restaurant chains. In terms of the quality of the solution per time expended, the simulated annealing heuristic was the top performer. We had expected that the IP models would perform better as their degree of detail increased. This had led us to believe that RevMgtIP would perform better than TimeIP, which in turn would perform better than NaïveIP. We were proven wrong since NaiveIP-A outperformed all IP solutions for the entire week problem and TimeIP had the best IP performance for the daily problem. Using paired *t*-tests of the difference between a model's RevPASH and the best possible RevPASH, SimAnneal-N and SimAnneal-S were statistically indistinguishable (at the 0.01 significance level), while SimAnneal-N was statistically superior to all other models.

The use of either simulated annealing heuristic requires access to a restaurant table simulator. If such a simulator is not available, our recommended heuristic is NaïveIP-A or TimeIP. These models yielded solutions within about 0.5% of optimal and were easily solved

with commercially available IP software. The size of these models, even in its weekly form, was such that we were able to solve them using Solver in Microsoft Excel®.

Of the three variants of RevMgtIP, the one using 5 min time periods (RevMgtIP5) performed best. We believe that it outperformed RevMgtIP15 since its 5 min time periods allow a better representation of the processing of parties in the restaurant. Conversely, 3 min time periods were not as effective as 5 min periods, because of the granularity of party demand that manifested itself with short-duration time periods (recall that RevMgtIP does not allow parties to wait for service-service must commence in the period in which the party arrives or the party will be lost). Also, the size of RevMgtIP3 model for the weekly problem resulted in it requiring more time to solve than SimAnneal. Although the performance of RevMgtIP5 was not quite as good as that of NaiveIP-A or TimeIP, the additional information available from its sensitivity analysis may make it an attractive solution approach.

Our results enabled us to evaluate the potential revenue gain associated with changing the table configuration from day to day compared to maintaining a static table configuration. The benefit of doing so was US$ 497 per week (US$ 45,219–US$ 44,722) or 1.1% of total revenue. Clearly, this would seem to be warranted especially since in most cases, additional labor would not be required. In addition, if the restaurant was configured with primarily two-tops, tables could be easily reconfigured on a daily basis. Given the different optimal table mix across days, an interesting design problem would address the most expedient way to reconfigure the restaurant. For example, if the restaurant contained primarily 2-tops, tables could be easily reconfigured on a daily basis.

The performance of the NaïveIP-A model surprised us. We had thought it would be the poorest performer of the methods tested because of its simplistic nature and lack of inclusion of data on party value, party duration and demand. Still, if such a simple model could lead to near optimal results, it warranted further investigation. We decided to perform a second, more broadly based experiment, which we describe below, to better benchmark the performance of the NaïveIP-A and NaïveIP-B model against SimAnneal-N. SimAnneal-N had developed optimal solutions for the weekly problem in the test problem and had developed a solution within 0.02% of optimal for the single day problem. Complete enumeration would always find the optimal solution, but the time involved rendered this approach impractical for our experiment.

5. Experiment 2

Experiment 2 contained 96 distinct restaurant scenarios that we constructed, varying on six environmental factors. In selecting the factors, and their levels, we drew upon the data from the actual restaurant used in Experiment 1 and upon our personal knowledge of the restaurant industry. In this experiment, we simulated 100 days of restaurant operation.

5.1. Factors tested

We tested a variety of factors, including the range of durations across party size, the variability of duration with party sizes, the demand intensity, party value differences, party size mix and restaurant size. We describe each below.

Duration range across party sizes. Our original data showed that in general, smaller parties took less time to dine than larger parties. We wanted to see if the difference between the shortest dining time and the longest dining time affected the results (with the mean held

constant). The experiment used two ranges of duration across party sizes: 20 min and 40 min. We expected that the SimAnneal and the Naïve-B models would outperform the Naïve-A model since they include information on dining duration.

5.1.1. Coefficient of variation in dining durations

Similarly, we thought the coefficient of variation of dining durations within party sizes might affect the results. Our original data had a coefficient of variation of approximately 0.4. We wanted to test how the models would react to a coefficient of variation of 0.3 and 0.5. Since the SimAnneal model explicitly considers the standard deviation of dining duration, we believed that it would outperform the Naïve models.

5.1.2. Demand intensity

We had already increased demand levels so that the restaurant was operating at capacity, but wanted to test the sensitivity of the models at even higher demand levels. We tested two levels of demand intensity: 100% and 120%. Again, we expected the SimAnneal model to outperform the Naïve models since it was the only model to explicitly consider demand levels.

5.1.3. Party value range

In the restaurant described in Experiment 1, smaller parties spent more per person than larger parties. We wanted to see if varying the difference between the smallest and largest party value would impact the results and so we tested two levels of party value range (low and high). We expected the SimAnneal-N to outperform the Naïve models since it explicitly considers party value.

5.1.4. Party size

The restaurant described in Experiment 1 had an average party size of about 2.5 people. We wanted to see how the heuristics reacted to an average party size of 2.5 and 3.5 people. In previous research, Thompson (2002) found that different party sizes can affect table mix results because of the necessity for larger tables. Since all models incorporated information on party size mix, we expected similar performance across party sizes.

5.1.5. Restaurant size

The original data set had 240 seats. For the test case, we wanted to evaluate the model sensitivity to a small (50 seat), medium (200 seat) and large (1000 seat) restaurant. Since all models incorporated information on restaurant size, we expected similar performance across the factor's levels.

In total, Experiment 2 contained 96 scenarios (Table 8). Since the Naïve-A model only considered two factors (party size mix and restaurant size), it yielded only 6 different table mixes across the 96 scenarios. The Naïve-B model considered three factors (party size mix, restaurant size and duration difference), so it yielded 12 different table mixes across the 96 scenarios. Since SimAnneal considered all six factors, it yielded 96 table mixes.

In Experiment 2 we allowed five different table sizes (2, 4, 6, 8 and 10). With these table sizes, the 50-, 200-, and 1000-seat restaurants had a total of 333, 44,559, and 22,849,600 possible table size combinations. Because of the large number of table combinations, we allowed the SimAnneal-N model to use up to 200 iterations to find the solution (as opposed to 100 iterations in the original study).

To evaluate the revenue performance of the Naïve models, we entered the table mix solution into the TableMix model along with the appropriate factor levels and ran the simulation to find the resulting revenues.

Since we had different restaurant sizes, we compared the models on the basis of revenue per available seat (RevPAS). In addition, we only studied the entire week problem and did not analyze differences by day of week.

5.2. Results

On average, the SimAnneal-N model performed slightly better than both the Naïve-A model (99.0% of SimAnneal-N) and the Naïve-B model (98.5% of SimAnneal-N). The Naïve-A model performed nearly as well as in our initial study (99.5%), and continued to outperform the Naïve-B model (Significant at the $P < 0.001$ level) (Table 9).

5.2.1. Duration range across party sizes

On average, the RevPAS was higher by US$ 1.99 for the higher duration range than for the lower range. The Naïve-A model achieved 98.8% of SimAnneal's RevPAS for the 20 min range and 99.2% for the 40 min range. The Naïve-B model achieved 98.4% of SimAnneal's RevPAS for the 20 min range and 98.5% for the 40 min range.

5.2.2. Coefficient of variation in dining durations

On average, the RevPAS was US$ 1.01 higher for the 0.3 level than for the 0.5 level. The Naïve-A model reached about 99% of SimAnneal for both cases, while the Naïve-B model achieved 98.2–98.7% of the SimAnneal results for both levels.

5.2.3. Demand intensity

Not surprisingly, the RevPAS was higher for the 120% level than for the lower level (by US$ 1.93). With higher demand intensity, the SimAnneal model has greater choice of parties to accept or reject. With excess demand, it is possible to accept more higher-value parties. The Naïve-A model results were 99.5% for the 100% case and 98.5% for the 120% case, while the Naïve-B model achieved 99.7% of the SimAnneal results for the 100% case and 97.3% for the 120% level. The Naïve models did not perform as well for higher demand intensities, which is not surprising since their formulations do not account for demand intensity.

5.2.4. Party value range

RevPAS was US$ 2.87 higher for the higher level of this factor than for the lower level. The Naïve-A model reached about 99% for both party value ranges while the Naïve-B model achieved 98.3–98.6% of the SimAnneal results.

5.2.5. Party size

The RevPAS was higher for the smaller mean party size, which is not surprising given the higher per person value of smaller party sizes. The Naïve-A model achieved about 99% of SimAnneal's RevPAS for both party sizes, while the Naïve-B model achieved about 98.5% of SimAnneal's RevPAS across party sizes.

5.2.6. Restaurant size

In general, the RevPAS was higher for the 1000-seat case than for the 200-seat and the 50-seat cases. The Naïve-A model achieved 98.8% of the SimAnneal results for the 1000-seat

restaurant and between 99.0% and 99.2% for the smaller restaurants. The Naïve-B achieved 99.1% of the SimAnneal for the 50-seat restaurant; 98.5% for the 200-seat restaurant; and 97.9% for the 1000-seat restaurant.

5.2.7. Table assignment rule (TAR)

To test the effect of the largest-waiting-party-that-fits TAR that we employed, we randomly sampled 10 restaurant scenarios from the 96 in Experiment 2. On these 10 samples, we ran SimAnneal using a FCFS (or longest-waiting party) table assignment rule. The FCFS rule averaged 97.57% of the RevPASH of the largest-waiting-party-that-fits TAR. These results further demonstrate the need for additional research on the effects of TARs.

5.3. Discussion

Overall, NaïveIP-A averaged within 1% of the SimAnneal solutions, while NaïveIP-B averaged within 1.5% of those solutions. Initially we were quite surprised that NaïveIP-A outperformed NaïveIP-B, particularly since NaïveIP-B incorporates information on dining duration by party size, while NaïveIP-A does not. However, the reason for this paradox became apparent. It relates to our experimental condition where smaller parties are more valuable, per person. Since smaller parties are more valuable, relatively more seats should be allocated to them. However, neither NaïveIP-A nor NaïveIP-B considers party value in their calculations. NaïveIP-B, by considering dining duration, actually allocates *more* seats towards larger parties than does NaïveIP-A, thus reducing its effectiveness. This leads to some suggestions for further enhancements to the NaïveIP models, which we identify below.

6. Conclusions

There is substantial room for improvement in how restaurant managers match their capacity to demand. The ability to quickly identify an optimal or near optimal table mix is likely to be of great relevance, given the size of the industry. For example, the optimal table mix for the restaurant described in Experiment 1 can allow the restaurant to increase revenue by over 32% (=US$ 44,722/US$ 33,712 × 100%) compared to its existing table mix. It is reasonable to ask why a restaurant would be operating with such a poor performing table mix. As in most businesses, capacity, in this case table mix, is taken as a given, and little thought is given to it once the restaurant is constructed. Clearly, this study indicates that the table mix has considerable revenue impact and is worthy of consideration.

We proposed and tested a variation of solution techniques and found that simulated annealing had the best performance, though the NaïveIP-A and TimeIP models had performance within 0.5% of optimal. We also found that optimizing table mix by day of rather than for the entire week led to a 1.1% increase in revenue.

In Experiment 2, we tested the robustness of the NaïveIP models under a variety of circumstances and found that NaïveIP-A performed within 1% of the Simulated Annealing solutions, while NaïveIP-B was within 1.5% of the Simulated Annealing solutions.

Clearly, this research is not without limitations. In Experiment 1, we did not evaluate any table mix with tables larger than eight seats and, in Experiment 2, we did not evaluate any table mix with tables larger than ten seats. Given the assumptions of our experiment, this meant that we did not serve any party larger than 8 (10) people. The main reason we limited our largest

table size was combinatorial complexity. For example in Experiment 1, allowing 10-top tables in addition to the 2-tops, 4-tops, 6-tops and 8-tops increases the number of table mixes to be evaluated to 88,759 or 6.55 times the number of table mixes we evaluated. Given the performance of the SimAnneal heuristics, however, we are confident that such heuristics could yield optimal or near-optimal performance even in an environment with this increased flexibility.

Once the optimal table mix is found, a number of operational concerns associated with a typically increased number of tables must be addressed. For example, staffing levels and materials requirements will probably increase and the increased flow of customers may result in bottlenecks developing in the kitchen or bar (Kimes, 2004). Other considerations are social norms on personal space (Hall, 1981) and how a restaurant's design can influences table configuration. Obviously, our research leaves many unanswered questions in restaurant design.

Given the rather impressive performance of the simplest models we investigate, we believe that further enhancements of the NaïveIP models are warranted. We recommend that some of the factors currently missing from the NaïveIP models be incorporated into the models. For example, enhancements to the models could address effect of demand intensity and differences in customer values across party sizes. In addition, research testing the impact of various table assignment rules on the NaiveIP models seems warranted.

Our investigation hints at the importance of linking capacity planning and revenue management research. To date, these areas have largely operated independently. A firm can only make good decisions about which customers to serve to if it has the right capacity to sell. While this research was confined to a full-service restaurant, we expect that our approaches to capacity planning and revenue management can also be applied to other service firms.

Table 1. Pseudo code of the simulated annealing heuristic.

1. Set Seats = 240, *temp* = 1, *cooling* = 0.95, *Iteration* = 0, *MaxIterations* = 100, *BestValue* = 0 and *IncumbentValue* = 0
2. Develop starting table mix (see narrative).
3. Add the current table mix to the list of mixes evaluated.
4. Set Iteration = *Iteration* + 1.
5. Simulate the current table mix, using the TABLEMIX simulation model. Let *CurrentValue* equal the value of the current mix.
6. Set *ReplaceIncumbent* = false.
7. If (*CurrentValue* > *BestValue*) then replace the best table mix with the current table mix.
8. If (*CurrentValue* ≥ *IncumbentValue*) then set *ReplaceIncumbent* = true and go to step 11; otherwise continue with step 9.
9. Set x = random[0, 1).
10. If (x < exp((*CurrentValue* − *IncumbentValue*)/temp)) then set *ReplaceIncumbent* = true.
11. If (*ReplaceIncumbent*) then replace the incumbent table mix with the current mix; otherwise replace the current table mix with the incumbent table mix.
12. Set *DropProp* = random [0.1, 0.2).
13. Randomly select tables and remove them from the current table mix. Repeat until at least *Seats* × *DropProp* seats have been removed from the mix.
14. Iteratively add tables to the mix until the current mix has a total of *Seats*. Select a table size (that would not result in *Seats* being exceed), where the probability of a table being selected is 1/(number of seats).
15. If the current table mix is the same as any that have already been evaluated, return to step 12.
16. If (modulus(*Iteration*, 2) = 0) then set *temp* = *temp* × cooling.
17. If (*Iteration* < *MaxIterations*) then return to step 3; otherwise return the best table mix.

Table 2. Mean dining duration (minutes), by party size and day of week.

Party size	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	40.5	38.3	43	35.7	40.6	48.3	42.9
2	45.5	47	47.8	47.5	47.9	47.4	46.4
3	48.4	53.4	52.6	52.9	49.7	53.7	50.5
4	51.1	52.6	52.8	56.7	53	55.7	54.9
5	54.5	62.3	60.3	59.5	66.1	55.4	55.2
6	69.9	67	61.8	66.3	59.8	63.5	61
7	68.7	73.5	83	74.1	58.8	59.8	62.4
8	62.6	76.2	89	59.5	92.5	74.8	72.2
9	64.3	51.4	61.2	67.8	59.8	83.2	71.8
10	64.3	51.4	61.2	67.8	59.8	83.2	71.8

Table 3. Mean party revenue, by party size and day of week

Party size	Sunday (US$)	Monday (US$)	Tuesday (US$)	Wednesday (US$)	Thursday (US$)	Friday (US$)	Saturday (US$)
1	22.02	17.9	19.11	16.22	16.78	33.18	29.6
2	27.94	30.01	29.12	27.48	29.39	29.98	28.78
3	36.12	39.13	37.9	38.01	37.03	41.54	39.47
4	47.2	47.45	44.42	52.52	48.93	49.21	46.97
5	56.38	62.49	55.05	61.92	68.55	56.32	61.16
6	76.08	72.75	58.08	75.61	78.52	73.03	74.7
7	79.21	74.07	42.15	92.38	73.94	75.72	86.97
8	84.65	98.55	115.25	100.27	137.87	86.27	93.3
9	132.14	102.79	86.96	111.46	94.05	137.88	131
10	123.14	116.11	99.42	126.79	128.33	122.73	127.73

Table 4. Solution times, in minutes, by day and week for the various methods.

Method	Solution time per single day[a]	Solution time per week[a]
Enum	139	973
NaïveIPA[b]	0.01	0.01
NaïveIPB[b]	0.02	0.02
TimeIP[b]	0.05	0.18
RevMgtIP15[b]	0.02	0.03
RevMgtIP5[b]	0.02	0.73
RevMgtIP3[b]	0.13	7.58
SimAnneal[c]	1.16	6.05
[a] On a Pentium IV 2.0 GHz personal computer.		
[b] Model solved using SAS-OR®.		
[c] To evaluate 100 table mixes.		

Table 5. Table mixes recommended by the solution methods.

Method	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Whole week
Enum	50-23-4-3	59-23-5-0	67-22-3-0	59-22-3-2	55-23-5-1	52-24-4-2	51-23-5-2	56-24-4-1
NaïveIP-A	46-22-6-3	57-20-5-2	64-20-4-1	59-21-5-1	55-23-5-1	51-23-5-2	49-24-5-2	53-22-5-2
NaïveIP-B	42-22-6-4	51-20-7-2	59-21-5-1	50-22-6-2	50-24-6-1	45-24-5-3	44-23-6-3	50-22-6-2
TimeIP	46-24-6-2	58-22-6-0	65-23-3-0	57-22-5-1	49-26-5-1	47-26-5-2	47-25-5-2	52-25-6-0
RevMgtIP15	47-20-7-3	54-24-6-0	62-21-4-1	52-19-6-3	51-25-5-1	47-26-3-3	46-25-4-3	48-23-6-2
RevMgtIP5	46-20-6-4	55-22-7-0	68-20-4-0	57-20-5-2	54-22-6-1	47-27-5-1	47-23-5-3	49-24-5-2
RevMgtIP3	45-22-5-4	54-24-6-0	67-20-3-1	57-17-7-2	51-22-7-1	50-25-4-2	47-23-5-3	50-22-6-2
SimAnneal-S[a]	49-22-5-3	59-23-5-0 (45)	67-22-3-0 (35)	59-22-3-2 (78)	55-23-5-1 (98)	55-26-3-1	51-23-5-2 (33)	56-24-4-1 (80)
SimAnneal-N[a]	50-23-4-3 (31)	59-23-5-0 (41)	67-22-3-0 (55)	59-22-3-2 (33)	57-22-5-1	52-24-4-2 (64)	51-23-5-2 (4)	56-24-4-1 (47)
Existing	2-56-2-0	2-56-2-0	2-56-2-0	2-56-2-0	2-56-2-0	2-56-2-0	2-56-2-0	2-56-2-0
[a] Values in parentheses are the iteration on which the heuristic identified the optimal solution								

Table 6. Revenue from the solution techniques' recommended table mixes.

Method	Sunday (US$)	Monday (US$)	Tuesday (US$)	Wednesday (US$)	Thursday (US$)	Friday (US$)	Saturday (US$)	Whole week (US$)	Single-day total (US$)	Single-day premium (%)[a]
Enum	6290	6601	6553	6332	6457	6539	6448	44722	45219	1.1
NaïveIP-A	6234	6539	6483	6324	6457	6513	6439	44545	44989	0.5
NaïveIP-B	6186	6405	6378	6181	6410	6373	6318	44207	44252	0.1
TimeIP	6249	6588	6536	6327	6422	6510	6420	44489	45052	1.3
RevMgtIP15	6250	6531	6457	6186	6448	6457	6386	43971	44715	1.7
RevMgtIP5	6237	6541	6544	6306	6449	6483	6391	44223	44950	1.6
RevMgtIP3	6241	6531	6526	6259	6406	6528	6391	44207	44882	1.5
SimAnneal-S	6286	6601	6553	6332	6457	6511	6448	44722	45188	1
SimAnneal-N	6290	6601	6553	6332	6448	6539	6448	44722	45211	1.1
Existing	4865	4829	4609	4683	4850	4962	4915	33712	33712	N/A

[a] Percentage increase in revenue yielded by the single-day best solutions, compared to the whole-week solution.

Table 7. Revenue from the solution techniques' recommended table mixes, expressed as a percentage of optimal revenue.

Method	Sunday (%)	Monday (%)	Tuesday (%)	Wednesday (%)	Thursday (%)	Friday (%)	Saturday (%)	Whole week (%)	Single-day total (%)
Enum	100	100	100	100	100	100	100	100	100
NaïveIP-A	99.1	99.1	98.9	99.9	100	99.6	99.9	99.6	99.49
NaïveIP-B	98.4	97	97.3	97.6	99.3	97.5	98	98.85	97.86
TimeIP	99.4	99.8	99.7	99.9	99.5	99.6	99.6	99.48	99.63
RevMgtIP15	99.4	98.9	98.5	97.7	99.9	98.8	99	98.32	98.89
RevMgtIP5	99.2	99.1	99.9	99.6	99.9	99.1	99.1	98.89	99.41
RevMgtIP3	99.2	98.9	99.6	98.9	99.2	99.8	99.1	98.85	99.26
SimAnneal-S	99.9	100	100	100	100	99.6	100	100	99.93
SimAnneal-N	100	100	100	100	99.9	100	100	100	99.98
Existing	77.3	73.2	70.3	74	75.1	75.9	76.2	75.4	N/A

Table 8. Experimental factors affecting each method.

Factor	Levels	NaïveIP-A	NaïveIP-B	SimAnneal
Duration range	2		X	X
Demand intensity	2			X
Party value range	2			X
Party size	2	X	X	X
Restaurant size	3	X	X	X
Coefficient of variation	2			X

Table 9. Results from Experiment 2.

Factor	Level	Method		
		SimAnneal (mean RevPAS) (US$)	Naïve-A (% of SimAnneal) (US$)	Naïve-B (% of SimAnneal) (US$)
Duration range across party sizes	20 min	27.68	27.36 (98.8%)	27.24 (98.4%)
	40 min	29.69	29.44 (99.2%)	29.25 (98.5%)
Coefficient of variation of dining duration	0.3	29.19	28.92 (99.1%)	28.82 (98.7%)
	0.5	28.18	27.87 (98.9%)	27.67 (98.2%)
Demand intensity	100%	27.72	27.59 (99.5%)	27.63 (99.7%)
	120%	29.65	29.21 (98.5%)	28.86 (97.3%)
Party value range	Low	27.25	27.00 (99.1%)	26.87 (98.6%)
	High	30.12	29.79 (98.9%)	29.62 (98.3%)
Party size	2.5 Customers	29.75	29.50 (99.2%)	29.33 (98.6%)
	3.5 Customers	27.61	27.29 (98.8%)	27.16 (98.4%)
Restaurant size	50 seats	27	26.78 (99.2%)	26.75 (99.1%)
	200 seats	28.98	28.68 (99.0%)	28.54 (98.5%)
	1000 seats	30.07	29.72 (98.8%)	29.43 (97.9%)
Overall		28.68	28.40 (99.0%)	28.24 (98.5%)

Figure 1. Party arrival rate by 15 min period.

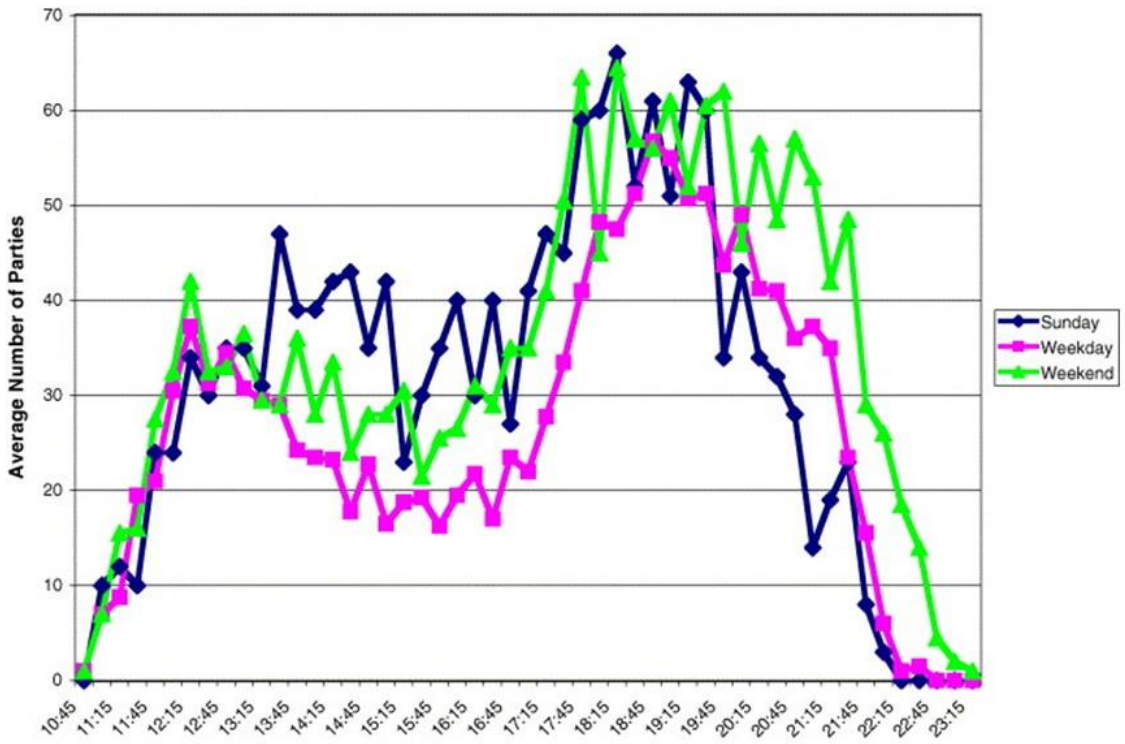


Figure 2. Party size mix by day of week.

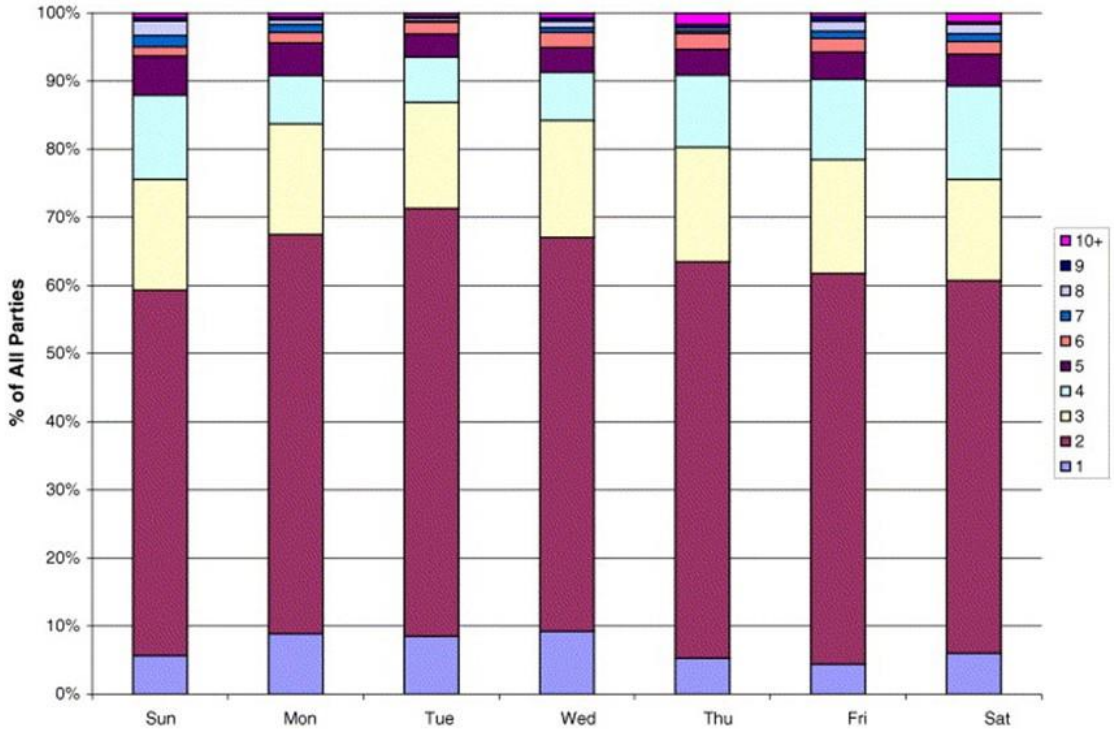


Figure 3. Seat occupancy by day of week and time of day.

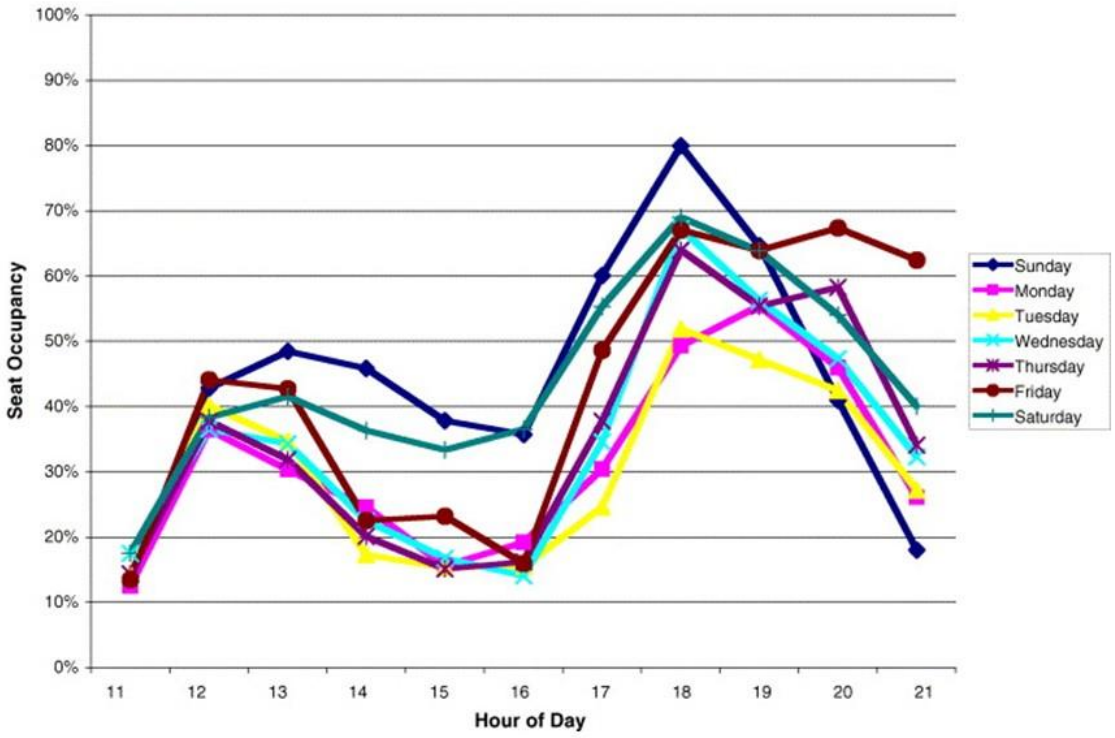
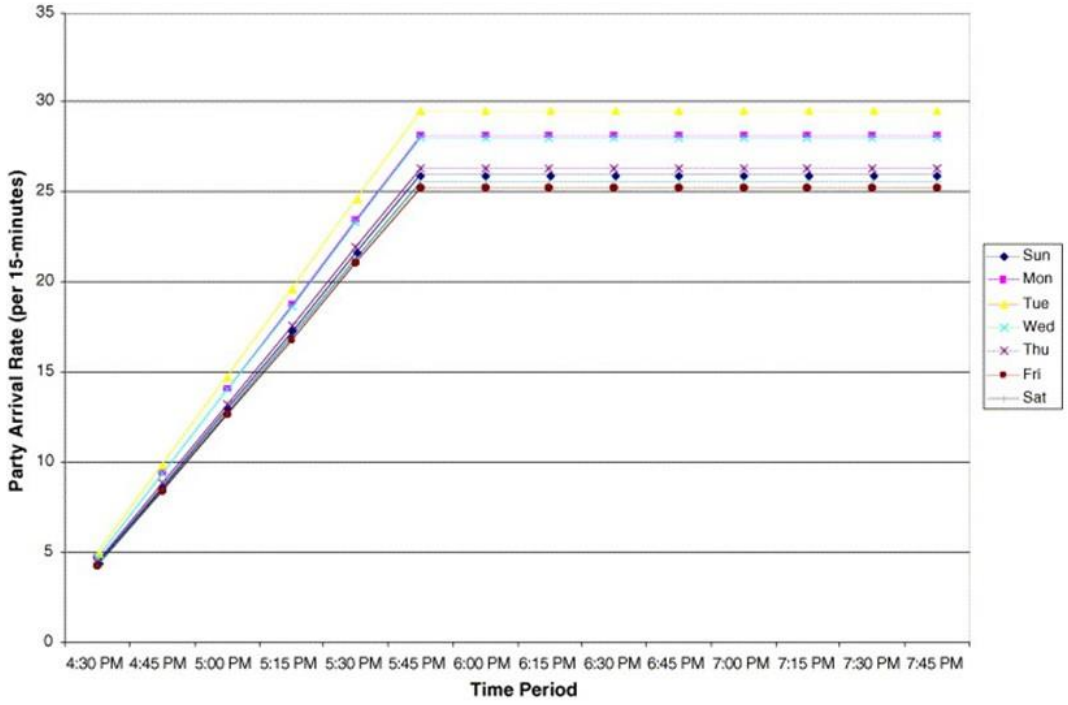


Figure 4. Daily party arrival rates, by 15 min period.



References

Baker, T., Collier, D.A., 1999. A comparative revenue analysis of hotel yield management heuristics. *Decision Sciences 30* (1), 239–263.

Baker, T., Murthy, N.N., Jayaraman, V., 2002. Service package switching in hotel revenue management systems. *Decision Sciences 33* (1), 109–132.

Banker, R.D., Morey, R.C., 1993. Integrated system design and operational decisions for service sector outlets. *Journal of Operations Management 11* (1), 81–98.

Belobaba, P.P., 1989. Application of a probabilistic decision model to airline seat inventory control. *Operations Research 37* (2), 183–197.

Bertsimas, D., Shioda, R., 2003. Restaurant revenue management. *Operations Research 51* (3), 472–486.

Bitran, G., Mondschein, S.V., 1997. Periodic pricing of seasonal products in retailing. *Management Science 43*, 64–79.

Brusco, M.J., Thompson, G.M., Jacobs, L.W., 1997. A morph-based simulated annealing heuristic for a modified bin-packing problem. *Journal of the Operational Research Society 48*, 433–439.

Carroll, W.J., Grimes, R.C., 1995. Evolutionary change in product management: experiences in the car rental industry. *Interfaces 25* (5), 84–104.

Choi, S., Kimes, S.E., 2002. The impact of distribution channels on revenue management. *Cornell Hotel and Restaurant Administration Quarterly 43* (3), 23–31.

Gallego, G., van Ryzin, G.J., 1994. A multi-product dynamic pricing problem and its application to network yield management. *Operations Research 45*, 24–41.

Geraghty, M.K., Johnson, E., 1997. Revenue management saves National Car Rental. *Interfaces 27* (1), 107–127.

Goodale, J.C., Verma, R.V., Pullman, M.E., 2003. A market utility-based model for capacity scheduling in mass services. *Production and Operations Management 12* (2), 165–185.

Hall, E.T., 1981. *The Silent Language*. Random House, New York.

Hanks, R.B., Noland, R.P., Cross, R.G., 1992. Discounting in the hotel industry, a new approach. *Cornell Hotel and Restaurant Administration Quarterly 33* (3), 40–45.

Kimes, S.E., 2004. Revenue management: implementation at Chevys Arrowhead. *Cornell Hotel and Restaurant Administration Quarterly 45* (1), 52–67.

Kimes, S.E., Barrash, D.I., Alexander, J.E., 1999. Developing a restaurant revenue-management strategy. *Cornell Hotel and Restaurant Administration Quarterly 34* (5), 1–30.

Kimes, S.E., Chase, R.B., Choi, S., Ngonzi, E.N., Lee, P.Y., 1998. Restaurant revenue management. *Cornell Hotel and Restaurant Administration Quarterly 40* (3), 40–45.

Kimes, S.E., Wirtz, J., 2002. Perceived fairness of demand-based pricing for restaurants. *Cornell Hotel and Restaurant Administration Quarterly 41* (1), 31–38.

Kimes, S.E., Wirtz, J., 2003. When does revenue management become acceptable? *Journal of Service Research 6* (2), 125–135.

Kimes, S.E., Wirtz, J., Noone, B.M., 2002. How long should dinner take?: measuring expected meal duration for restaurant revenue management. *Journal of Revenue and Pricing Management 1* (3), 220–233.

Kimes, S.E., Thompson, G.M., 2004. Restaurant revenue management at Chevys: determining the best table mix. *Decision Sciences 35 (*3), 371–392.

Klassen, K.J., Rohleder, T.R., 2001. Combining operations and marketing to manage capacity and demand in services. *The Service Industries Journal 21* (2), 1–30.

Lieberman, W.H., Dieck, T., 2002. Expanding the revenue management frontier: optimal air planning in the cruise industry. *Journal of Revenue and Pricing Management 1* (1), 7–24.

Lovelock, C.H., 1992. Strategies for managing capacity constrained service organizations. In: Lovelock, C.H. (Ed.), *Managing Services: Marketing, Operations and Human Resources*. 2nd ed. Prentice-Hall, New Jersey, pp. 154–168.

McGill, J.I., van Ryzin, G.J., 1999. Revenue management: research overview and prospects. *Transportation Science 33*, 233–256.

Ng, I.C.L., Wirtz, J., Lee, K.S., 1999. The strategic role of unused service capacity. International *Journal of Service Industry Management 10* (2), 211–238.

National Restaurant Association, 2002. 2002 Restaurant Industry Profile. http://www.restaurant.org/research/forecast_overview.cfm.

Rexing, B., Barnhart, C., Kniker, T., Jarrah, A., Krishnamurthy, N., 2000. Airline fleet assignment with time windows. *Transportation Science 34* (1), 1526–5447.

Robson, S., 2004. Personal communication.

Sasser, E.E., 1976. Match supply and demand in service industries. *Harvard Business Review* 133–140.

Sill, B., 1991. Capacity management: making your service delivery system more productive. *Cornell Hotel and Restaurant Administration Quarterly 33* (1), 77–87.

Smith, B.A., Leimkuhler, J.F., Darrow, R.M., 1992. Yield management at American Airlines. *Interfaces 22* (1), 8–31.

Thompson, G.M., 1996. A simulated-annealing heuristic for shift scheduling using non-continuously available employees. *Computers and Operations Research 23* (3), 275–288.

Thompson, G.M., 1995. Labor scheduling using NPV estimates of the marginal benefit of additional labor capacity. *Journal of Operations Management 13* (1), 67–86.

Thompson, G.M., 2002. Optimizing a restaurant's seating capacity: use dedicated or combinable tables. *Cornell Hotel and Restaurant Administration Quarterly 43* (4), 48–57.

Thompson, G.M., 2003. Optimizing restaurant-table configurations: specifying combinable tables. *Cornell Hotel and Restaurant Administration Quarterly 44* (1), 53–60.

Weatherford, L.R., Bodily, S.E., 1992. A taxonomy and research overview of perishable-asset revenue management: yield management, overbooking and pricing. *Operations Research 40* (5), 831–844.