IE 519 Dynamic Programming Term Project: Restaurant Revenue Management

Haedong Kim *The Department of Industrial and Mnufacturing Enginnering*

For the term project of the Dynamic Programming course, I have worked on restaurant revenue management of allocating customers to tables to maximize the revenue of the restaurants. Unlike the previous study on the similar settings, multiple sizes of parties and tables are considered. Probability distributions to model the multiple sizes of parties and tables are proposed. Then the dynamic programming model and seating schemes using that model are proposed to maximize the restaurant revenue.

Keywords: Revenue Management, Restaurant Revenue Management, Dynamic Programming, Guests Placement Scheme

Current version is 12/06/2018, and the replication files are available on the author's Github account (https://github.com/haedong31/dp_project).

1. Introduction

The objective of revenue management, also known as yield management, is to maximize revenue by providing the right product to the right customers at the right price and time without compromising customer satisfaction (Kimes, 1989; Netessine and Shumsky, 2002). Although the main research area of revenue management is pricing, it is not only limited to pricing but also includes it also includes optimizing inventory capacity or duration of service time (Donaghy, McMahon and McDowell, 1995). Revenue management began at the U.S. airline industry since the deregulation that allows airline companies the flexibility at pricing seats and controlling seat inventory (Belobaba, 1987). Revenue management has brought enormous successes in the airline industry. It is reported that in the early 1990s that American Airline estimated the annual benefit of yield management at $500 million (Smith, Leimkuhler and Darrow, 1992). After the successes of the airline industry, other industries have adopted revenue management techniques including but not limited to hotels, automobile rentals, and cruise lines (McGill and Van Ryzin, 1999).

In this study, we focus on the application of revenue management to restaurants. Restaurant revenue management, RRM for short, is pioneered by Kimes et al. (1998) and Kimes et al. (1999). Kimes et al. (1998) suggests the four characteristics that make revenue management more effective if some operation has them: relatively fixed capacity, predictable demand, perishable inventory, and appropriate cost and pricing structure. The authors argue that the restaurant industry has such features. Specifically, they claim that what perishable in restaurants is available time for tables to serve customers, instead of raw food. Therefore, they come up with the measure named revenue per available seat hour (RevPASH) to check the performance of restaurants. Recently, other mesaures taking into account margin of food and cost of raw food are presented (Heo, 2017). In Kimes et al. (1999), the five steps to implement RRM are described. A way to manage meal duration is also explored (Kimes, Wirtz and Noone, 2002). These methods were testes in real-world restarant operations (Kimes and Wirtz, 2003; Kimes, 2004). While these studies provide general principles of RRM, they lack quantitatevely rigorous analysis.

There are relatively few papers pertaining to quantitative RRM. The first research to mathematically model the restaurant operation has been done in Bertsimas and Shioda (2003). In Bertsimas and Shioda (2003), for seating policies, an approximate dynamic programming formulation based on interger programming for restaurants do not accept booking is proposed, and for restaurants with reservation, a stochastic gradient algorithm is described. Seating considers how to place guests to the right table to maximize renvenue. Anotheor research field of RRM is table mix concerning how to constitute a restaurant with how many and what type of tables (Kimes and Beard, 2013). Table mix along with seating without reservation is quantitatively analyzed in Guerriero, Miglionico and Olivito (2014).

Both of Bertsimas and Shioda (2003) and Guerriero, Miglionico and Olivito (2014) use approximate dynamic programming to solve RRM problems. Since decisions in the restaurant operation are made sequentially by time, dynamic programming is a right choice to model it. However, because of the curse of dimensionality, the both studies leverage interger programming and linear programming respectively to approximate a value function in a dynamic programming formulation. In contrast, Kuo (2010) uses an exact dynamic programming model, so that accurately evaluates marginal utility of preserving a table by rejecting arrived party for a chance of bigger group arriving in the future. However, this study simplifies operation in many aspects. For example, the author assumes that there are only two sizes of parties and tables do not have a limitation so it can serve the both classes of guests. Therefore, in this paper, we propose an exact dynamic programming model that consider more realistic situations in restaurants. Specificially, we consider several sizes of parties and tables.

This paper is structured as follows. Section 2 introduces the specific problem statement including the dynamic programming formulations. Section 3 describes the solution procedure to generate seating policies. Section 4 shows a computational experiments using artificial data. Last, Section 5 presents the conclusions and suggestions for future research.

2. Problem Statement

Usually, a restaurant consists of different dimensions of tables to serve various sizes of customers. During the operation of a restaurant, which has multiple sizes of tables, a floor manager has to make following decisions upon the arrival of a party. First, a manager has to decide whether to accept the party or turn down them. If she/he chooses to receive the arrived guests, now a manager needs to determine which table to serve the customers. Revenue may increase by rejecting the arrived party to preserve a table for the chance of having a larger group in the future that generate bigger revenue than smaller parties. In addition, since there are multiple classes of tables, a manager must be careful to determine in which table to serve the party to maximize revenue. Therefore, this project develops a dynamic programming model that can be used to generate a seating scheme to increase revenue.

A set of assumptions are made to model the restaurant operation. It is assumed that table sizes are only even numbers for simplicity, and tables cannot be merged or divided. Similar to Kuo (2010), the time horizon is discretized by sufficiently small enough periods so that only one party arrives at a period. Arrivals of customers and the size of them for a given period are random events, so a modified Poisson distribution is used to model the arrivals. State space is represented as a tuple that has the length of the number of table sizes, so each element indicates the number of occupied tables of each size.

At the end of a period, the state will change according to the departures of parties have done their meals. The departures occur probabilistically, and a binomial distribution is used to model it. It is assumed that departure probability is the same across all sizes of parties, while it is possible to set up the different probability for sizes, such as a larger group has a lower probability than a smaller group for simplicity.

2.1. Assumptions

- Operation hours are discretized into multiple small periods by some unit time.
- For a given period, customer arrivals occur at most once at the beginning of the period.
- Operation hours can be discretized by different unit times. For example, lunch and evening hours may be discretized finer than others so more parties arrive during that hours.
- Size of parties has an upper bound
- The restaurant has a table large enough to serve the largest party
- Overbooking is not allowed so that the number of occupied tables cannot exceed fixed capacity
- Tables cannot be merged or splitted.
- The probability of a table becoming available is same for all sizes of tables.
- The restaurant does not accept reservations.

- Queueing up is not allowed.

2.2. Data

- $n := \{1, 2, ..., N\}$ period
- $p := \{0, 1, ..., P\}$ party size.
- $t := \{1, 2, ..., T\}$ table size, and $T \geq P$.
- $R_p := rp$ the revenue by a party of size p, which is a linear function of p with a coefficient r.
- $C_t :=$ the total number of tables (i.e. capacity) of size t.
- $q :=$ the probability of a table becoming available (i.e., a party leave the restaurant).
- $\alpha :=$ a penalty factor for wating table capacity
- $\lambda :=$ the average size of arriving parties.

2.3. Probability Distribution of Arrivals

Let A be the random variable of size of an arriving party for a period. If size is 0, it means there is no arrivals for that period. In this paper, A is modeled by a modified Poission distribution having p as a domain by adjusting the constant so that the probability distribution sums up to 1. Hence, its probability mass function is

$$P(A = p) = \frac{P!}{e^{\lambda}\Gamma(P+1, \lambda)} \frac{\lambda^p}{p!},$$

where Γ is the incomplete gamma function defined by

$$\Gamma(s, x) = \int_x^{\infty} t^{s-1} e^{-t} dt.$$

We interpreate the number of arrivals for a given period as size of a party. The advantage of this approach is that we can control the tendency of arrival sizes by changing the parameter of A, λ. It seems resonable to increase λ for the periods of evening hours to reflect that large parties more likely arrive in the evening.

2.4. State Space

For period n, let X_n be the tuple of the length of the number of table sizes. The first element of X_n is x_1 that is the number of occuipied tables of size 1, the second element x_2 is for the number of occuipied tables of size 2, and so forth. Furthermore, each element of X_n is bounded by corredponding capacity. In other words,

$$X_n := (x_1, x_2, ..., x_T) \leq (C_1, C_2, ..., C_T).$$

2.5. State Transformation

At the end of each period, parties leave the restaurant with probability q, which causes the state transformation, so we need to model the departures of customers to express the state transformation.

A party of an arbitrary size p finishes their meals and leave the restaurant with a probability q. It means that the probabilities of the departures of customers are the same as q regardless of part sizes, so the probabilities of arbitrary sized tables becoming available are also q. The Binomail distribution is a natural choice for the random variable D_t that is the number of the depatures of parties have occupied size t tables out of x_t at the end a period. Its probability mass function is

$$P(D_t = d_t) = \binom{x_t}{d_t} q^{d_t}(1-q)^{x_t - d_t}.$$

Since we deals with multiple sizes of tables, we need a joint probability

$$P(D_1 = d_1, D_2 = d_2, ..., D_T = d_T)$$

to model all the becoming available tables at the end of a period. It is resonable to assume that D_t's are independent each other, because customers usually work on their meals independently and do not have effects on other's meal duration. Therefore, the joint probability is

$$P(D_1 = d_1, D_2 = d_2, ..., D_T = d_T) = \prod_{t=\{1,...,T\}} P(D_t = d_t) = \prod_{t=\{1,...,T\}} \binom{x_t}{d_t} q^{d_t}(1-q)^{x_t - d_t},$$

which is the probability of X_n becoming X_{n+1} such that

$$X_{n+1} = (x_1 - d_1, x_2 - d_2, ..., x_T - d_T).$$

2.6. Action Space

Action space in the stated restaurant operations consists of two stages. First, it has to be decided whether to accept or refure the customer request. Next, a floor manager needs to determine at which tale to serve the party. These actions are only possible when there are available table large enough to serve the arrived party, and if the arrival occurs.

If there are no tables large enough to serve the party, then only options for the manager is to reject the request of the party. In addition, if the arrival does not occur, the restaurant has no options to chose.

2.7. Optimality Equation

Crucial parts of a dynamic progrraming model, the optimal value function, recurrence relation, and boundary conditions, are presented in this section. Let $V_n(X_n)$ be the maximum revenue from the beginning of period n to the end of the last period, N. Then the recurrence relation for the backward dynamic programming formulations is:

$$\begin{aligned} &V_n(X_n) = \\ &\sum_{p=1}^{P} P(A=p) \max \begin{cases} \max_u \{R_p + \alpha(u-p) + \sum_{d_T=0}^{x_T} \cdots \sum_{d_1=0}^{x_1} P(D_1 = d_1, ..., D_T = d_T) V_{n+1}(X'_{n+1})\} \\ \sum_{d_T=0}^{x_T} \cdots \sum_{d_1=0}^{x_1} P(D_1 = d_1, ..., D_T = d_T) V_{n+1}(X''_{n+1}) \end{cases} \\ &+ P(A=0) \sum_{d_T=0}^{x_T} \cdots \sum_{d_1=0}^{x_1} P(D_1 = d_1, ..., D_T = d_T) V_{n+1}(X''_{n+1}), \end{aligned}$$

where, for each p,

$$u \in \{t : t \geq p,\ x_t < C_t\},$$

$$X'_{n+1} = (x_1 - d_1, ..., x_u + 1 - d_u, ..., x_T - d_T),$$

$$X''_{n+1} = (x_1 - d_1, ..., x_T - d_T).$$

Interpretation of the recurrence relation is as follow. At the beginning of a period, a floor manager first chose to whether accept or not the arrived party, and if he/she accept the party, then the follow-up decision is table size to serve the customers. At the end of the period, some guests leave the restaurant. The expected maximum revenue of the given period is the sum of immediate revenue according to the decision structre stated earlier and the future expected maximum revenue.

Boundary conditions are need for the case when there is not available table for arrived party of size p. $u = \emptyset$ in this case. Then, in the reccrence relation,

$$\max \begin{cases} \max_u \{R_p + \alpha(u-p) + \sum_{d_T=0}^{x_T} \cdots \sum_{d_1=0}^{x_1} P(D_1 = d_1, ..., D_T = d_T) V_{n+1}(X'_{n+1})\} \\ \sum_{d_T=0}^{x_T} \cdots \sum_{d_1=0}^{x_1} P(D_1 = d_1, ..., D_T = d_T) V_{n+1}(X''_{n+1}) \end{cases}$$
$$= \sum_{d_T=0}^{x_T} \cdots \sum_{d_1=0}^{x_1} P(D_1 = d_1, ..., D_T = d_T) V_{n+1}(X''_{n+1}).$$

At the beginning of last period, the restaurant always accept a request of any party regardless of its size without the penalty of seating a party at a bigger table if there are available tables. Unless, the restaurant has no choice but to turn down the request. To model this senario, an indicator function $\mathbb{I}(p, X_N)$ is introduced to make revenue 0 when there are no available tables for party size p at period N. $\mathbb{I}(p, X_N)$ is 1 if there exist $t \in \{2, 3, ..., T\}$ such that $t \geq p$ and $x_t < C_t$ for $x_t \in X_N$, otherwise 0. Then,

$$V_N(X_N) = P(A = p) R_p \mathbb{I}(p, X_N).$$

3. Solution Procedure

A seating scheme may be getting complex as the number of party sizes and table sizes we consider become bigger. However, some general principles are available for any situation. First, we do not reject the largest size of party when if are available tables for them, because they guarantee large revenue. Second, if there are no available tables which are larger than or equal to the size of the arrived party, we have no choice but to turn down their request. Last, if we have a table just match with the size of the arrival, we assign the party to that table. Otherwise, seating decisions are depending on the levels of occupied tables. Therefore, our objective is to find the limits of the table levels so that decisions on accepting or rejecting arrival are made accordingly.

We developed the seating schemes for two models: 1) two party and table sizes (i.e., large and small), 2) three party and table sizes (i.e., large, medium, and small). For each case, a seating policy for each size of party except the largest one is presented. Abbreviations S, M, and L are uses to refering to small, medium, and large respectively.

3.1. *Case 1.* $p = \{S, L\}$ *and* $t = \{S, L\}$

For a large party, we always accept their request if $x_L < C_L$. For a small party, three states are possible:

1. $x_S < C_S$
2. $x_S = C_S$ and $x_L = C_L$
3. $x_S = C_S$ and $x_L < C_L$

In state 1, a floor manager always accept the small party and reject them in state 2. In state 3, decision vary according to the level of x_L. A floor manager may want to seat a small party to a large table if there are many idle large tables but to reject the small party if there are only a few of large tables to keep them for arivals of large parties in the future. To come up with a seating scheme for state 3, we need a marginal utility function evaluating the profit of preserving a large table for each period n. It is given by

$$\Delta_n(x_L) = V_n(C_S, x_L) - V_N(C_S, x_L + 1).$$

It should be noted that the utility functions defined this way are increasing functions, because the more occupied tables, the higher revenue by preserving a table. The table level limit for a small party $b_{n,S}$ is

$$b_{n,S} = \min\{x_L : \Delta_{n+1}(x_L) > R_L\}.$$

Therefore, at period n and state 3, the restaurant accepts a small party if the number of occupied large tables at the beginning of period n is less than $b_{n,S}$, $x_L < b_{n,S}$.

3.2. Case 2. $p = \{S, M, L\}$ and $t = \{S, M, L\}$

Similar to the Case 2, we always accept a large group if $x_L < C_L$ and reject otherwise. A seating scheme for a medium party is actually the same with the Case 1, because there is only one larger table size availabe for the medium party. Therefore, there are three states:

1. $x_M < C_M$
2. $x_M = C_M$ and $x_L = C_L$
3. $x_M = C_M$ and $x_L < C_L$

The restaurant always accept the party at state 1 and reject at state 2. A marginal utility function and a control limit are

$$\Delta_n^M(x_L) = \sum_{x_L} V_n(x_L, C_M, x_L) - V_N(x_L, C_M, x_L + 1),$$
$$b_n^M = \min\{x_L : \Delta_{n+1}(x_L) > R_M\}.$$

The only difference is we marginalize out x_L when calculating $\Delta_n^M(x_L)$, because x_L does not affect decisions on seating of medium groups. Therfore, the seating policy is let a medium party have a seat at a large table if $x_L < b_n^M$ for each period n.

A seating policy for a small party becomes complex because it is possilbe to seat them at two sizes of tables, medium and large. Five states are possible:

1. $x_S < C_S$
2. $x_S = C_S,\ x_M = C_M,\ x_L = C_L$
3. $x_S = C_S,\ x_M < C_M,\ x_L = C_L$
4. $x_S = C_S,\ x_M = C_M,\ x_L < C_L$
5. $x_S = C_S,\ x_M < C_M,\ x_L < C_L$

The similar way of analysis works for from state 1 to state 4. In state 1, we always accept a small party and reject in state 2. For state 3 and 4, marginal utility functions are

$$\Delta_{n,3}^S(x_M) = V_n(C_S,\ x_M,\ C_L) - V_n(C_S,\ x_M + 1,\ C_L),$$
$$\Delta_{n,4}^S(x_L) = V_n(C_S,\ C_M,\ x_L) - V_n(C_S,\ C_M, x_L + 1),$$

and the corresponding control limits are

$$b_{n,3}^S = \min\{x_M : \Delta_{n+1,3}^S(x_M) > R_S\},$$
$$b_{n,4}^S = \min\{x_L : \Delta_{n+1,4}^S(x_L) > R_S\}.$$

Therefore, if a small party arrives at state 3 and $x_M < b_{n,3}^S$, the restaurant accept the party. If the party arrivaes at state 4 and $x_L < b_{n,4}^S$, the restaurant accept their request.

The most complex situation is when the restaurant is in state 5. Now a seating scheme has to consider not only just one level of tables, but two levels simultaneously. At period n, for each level of x_M and x_L, we need to evaluate the revenue of preserving a medium table and a large table defined by $\Delta_{n,5M}^S(x_M, x_L) := V_n(C_S, x_M, x_L) - V_n(C_S, x_M + 1, x_L)$ and $\Delta_{n,5L}^S(x_M, x_L) = V_n(C_S, x_M, x_L) - V_n(C_S, x_M, x_L + 1)$. Then, we find a boundary value of preserving a medium or large table yield the higher revenue by increasing x_L and x_M. The search algorithm starts from when $x_L = 0$and $x_M = 0$. Then it checks that $\Delta_{n,5L}^S(x_M, x_L) < R_L$ and $\Delta_{n,5M}^S(x_M, x_L) < R_L$ increase x_M by 1. If the algorithm finds the first (x_L, x_M) such that $\Delta_{n,5L}^S(x_M, x_L) < R_L$, save (x_L, x_M) at $b_{n,5L}^S$ and it stops checking $\Delta_{n,5L}^S(x_M, x_L) < R_L$. Note that now the control limit $b_{n,5L}^S$ is not a constant but a vector. The same procedure is applying to $\Delta_{n,5M}^S(x_M, x_L) < R_L$ to find (X_L, x_M) satisfying that inequality. If the both coordinates have been found for $x_L = 0$, then do the same procedure for $x_L = 1$ and so forth. It looks like in pseudo code as follow:

```
b_L=list()
b_M=list()
for xL in range(1, C_L-1):
  for xM in range(1, C_M-1):
    if delta_L(xL, xM) > RS:
      # Skip it when we find the fist (xL, xM) satisfies delta_L(xL, xM) > RS
      b_L.append((xL, xM))
    if delta_M(xL,xM) > RS:
      # Skip it when we find the fist (xL, xM) satisfies delta_M(xL, xM) > RS
      b_M.append((xL, xM))
    if delta_L(xL,xM) > RS && delta_M(xL,xM) > RS:
        # Go to the next xL
```

where xL, xM, delta_L, delta_M, b_L, and b_M, are corresponding to x_L, x_M, $\Delta^S_{n,5L}$, $\Delta^S_{n,5M}$, $b^S_{n,5L}$, and $b^S_{n,5M}$.

A seating policy can be derived from $b^S_{n,5L}$ and $b^S_{n,5M}$. Upon arrival of a small party, check that there is an element of $b^L \in b^S_{n,5L}$ and $b^M \in b^S_{n,5L}$ such that $b^L \leq (x_L, x_M)$ and $b^M \leq (x_L, x_M)$. If they are true, rejecting the arrived party may yields the hiher revenue in the future. On the other hand, there if one of the inequalities does not hold, we may want to accept the party. For example, if there is $b^M \leq (x_L, x_M)$, but b^L such that $b^L \leq (x_L, x_M)$ does not exist, assigning the small party to a large party would yield the higher revenue than rejecting them. If both inequalities do not hold, accepting the party to a smaller table, mideum one.

4. Conclusions and Future Research

We proposed a dynaic programming model and seating schemes bases on that model. Although it seems to give promising results, this study lacks of simulation or experiments based on real data to verify the proposed algorithms. In addition, since generating schemes is getting complex and difficult as the numbers of sizes of parties and tables become larger, a general way to create seating policies is needed. Furthermore, considering waiting lines must be interesting and practical research.

References

Belobaba, Peter P. 1987. "Survey Paper—Airline yield management an overview of seat inventory control." *Transportation science* 21(2):63–73.

Bertsimas, Dimitris and Romy Shioda. 2003. "Restaurant revenue management." *Operations research* 51(3):472–486.

Donaghy, Kevin, Una McMahon and David McDowell. 1995. "Yield management: an overview." *International journal of hospitality management* 14(2):139–150.

Guerriero, Francesca, Giovanna Miglionico and Filomena Olivito. 2014. "Strategic and operational decisions in restaurant revenue management." *European Journal of Operational Research* 237(3):1119–1132.

Heo, Cindy Yoonjoung. 2017. "New performance indicators for restaurant revenue management: ProPASH and ProPASM." *International Journal of Hospitality Management* 61:1–3.

Kimes, Sheryl E. 1989. "The basics of yield management." *Cornell Hotel and Restaurant Administration Quarterly* 30(3):14–19.

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

Kimes, Sheryl E, Brian Sill, Robert Decker, Bill Quain, Michael W Sansbury, Stephen M LeBruto, Judy A Siguaw, Anna Mattila, Jon R Austin, Stephanie KA Robson et al. 1999. "2. Implementing restaurant revenue management: A Five-step approach." *The Cornell Hotel and Restaurant Administration Quarterly* 40(3):3–96.

Kimes, Sheryl E and Jochen Wirtz. 2003. "Revenue management at Prego Italian restaurant." *Asian Case Research Journal* 7(01):67–87.

Kimes, Sheryl E, Jochen Wirtz and Breffni M Noone. 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, Sheryl E and Jonathan Beard. 2013. "The future of restaurant revenue management." *Journal of Revenue and Pricing Management* 12(5):464–469.

Kimes, Sheryl E, Richard B Chase, Summee Choi, Philip Y Lee and Elizabeth N Ngonzi. 1998. "Restaurant revenue management: Applying yield management to the restaurant industry." *Cornell Hotel and Restaurant Administration Quarterly* 39(3):32–39.

Kuo, Wan-Ting. 2010. Application of Dynamic Yield Management to a Restaurant Seating Scenario. Master's thesis The Pennsylvania State University.

McGill, Jeffrey I and Garrett J Van Ryzin. 1999. "Revenue management: Research overview and prospects." *Transportation science* 33(2):233–256.

Netessine, Serguei and Robert Shumsky. 2002. "Introduction to the theory and practice of yield management." *INFORMS transactions on education* 3(1):34–44.

Smith, Barry C, John F Leimkuhler and Ross M Darrow. 1992. "Yield management at American airlines." *interfaces* 22(1):8–31.