Finding an optimal seating chart

Meghan L. Bellows[1] and J. D. Luc Peterson[2]
[1]*Department of Chemical and Biological Engineering, Princeton University*
[2]*Princeton Plasma Physics Laboratory, Princeton University*

EDITOR'S NOTE: The authors submitted this article long prior to the wedding. Somehow we managed to not see it until long after. The only benefit being that the article can and does include photographs taken at the wedding.

Introduction

Every year, millions of brides (not to mention their mothers, future mothers-in-law, and occasionally grooms) struggle with one of the most daunting tasks during the wedding-planning process: the seating chart. The guest responses are in, banquet hall is booked, menu choices have been made. You think the hard parts are over, but you have yet to embark upon the biggest headache of them all. In order to make this process easier, we present a mathematical formulation that models the seating chart problem. This model can be solved to find the optimal arrangement of guests at tables. At the very least, it can provide a starting point and hopefully minimize stress and arguments.

Model

The model describes the problem by maximizing the number of connections a particular guest has at their table. In other words, the model will seek to place guests who all know each other at the same table. A user must provide the total number of guests (*m*), the total number of tables (*n*), the total number of people each table can seat (*a*), the minimum number of people each guest should know at their table (*b*), and a connection matrix describing which guests know each other (C^{jk}). Table 1 provides descriptions of all the sets, variables, and parameters used in the model.

Set	**Description**
i	table number, from 1 to *n*, where *n* is the maximum number of tables
j, k	guest number, from 1 to *m*, where *m* is the maximum number of guests
Variable	**Description**
g_i^j	= 1 if guest *j* is seated at table *i*, 0 otherwise
g_i^k	= 1 if guest *k* is seated at table *i*, 0 otherwise
$p_i^{jk} = g_i^j g_i^k$	= 1 if guest *j* AND guest *k* are seated at table *i*, 0 otherwise

Parameter	Description
a	maximum number of guests a table can seat
b	minimum number of people each guest knows at their table
C^{jk}	connection matrix indicating relation of guest *j* to guest *k* (number from 0 to 50)

Table 1: Model Definitions

The model itself is written mathematically as:

$$\max_{g_i^j, g_i^k} \quad \sum_{i=1}^{n}\sum_{j=1}^{m-1}\sum_{k=j+1}^{m} C^{jk} p_i^{jk} \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} g_i^j = 1, \quad \forall j \tag{2}$$

$$\sum_{j=1}^{m} g_i^j \le a, \quad \forall i \tag{3}$$

$$\sum_{j=1}^{m} p_i^{jk} \ge (b+1) g_i^k, \quad \forall i,k,C^{jk} > 0 \tag{4}$$

$$\sum_{j=1}^{m} p_i^{jk} \le a g_i^k, \quad \forall i,k \tag{5}$$

$$\sum_{k=1}^{m} p_i^{jk} \le a g_i^j, \quad \forall i,j \tag{6}$$

$$g_i^j, g_i^k, p_i^{jk} \in \{0,1\} \tag{7}$$

In plain English, the model seeks to maximize the number of people each guest knows at their table (Eq. 1), subject to the fact that each guest can only sit at one table (Eq. 2), each table can only accommodate a fixed number of guests (Eq. 3), and each guest must know a minimum number of other guests at their table (Eq.4). Equations 5 and 6 are Eq. 3 multiplied by g_i^k and g_i^j, respectively (where $p_i^{jk} = g_i^j g_i^k$). These are introduced in order to linearize the model (which makes it much easier to solve). The final constraint (Eq. 7) defines g_i^j, g_i^k, and p_i^{jk} as binary variables, meaning they only equal zero or one. Table 1 defines the symbols in the model.

Once the sets have been defined (total number of tables and total number of guests) and the parameters have been set (total number of guests per table, minimum number of people each guests knows at their table, and the guest connection matrix), the model can be solved using mixed-integer linear optimization techniques to find the global optimal solution.

Example Problems

As an example, take a small wedding consisting only of the immediate family. There are 17 guests total (assume the bride and groom sit at their own sweetheart table) and since each table can seat 10 guests, only two tables are necessary. Let us also say that each guest must know at least one other people at their table. So $m = 17$, $n = 2$, $a = 10$, and $b = 1$.

There are $2^{17} = 131{,}702$ possible combinations for seating these guests. The 17 guests consist of 9 of the bride's guests and 8 of the groom's guests. Their names and relations are illustrated in Table 2. Guests have been assigned a number, which will be their *j* and *k* designation in the model. Based on this guest list the connection matrix can be constructed. Assume that all members of the bride's family know each other (assign a value of 1) and all members of the groom's family know each other. Assume that the bride's family has not met the groom's family yet (assign a value of 0). If two guests are married/engaged/dating/etc, assign a value of 50. We also feel that Abby and Martha (cousins) should sit at the same table, so they have been assigned a value of 10. Table 3 shows the full connection matrix C^{jk}.

Using GAMS software [1] with the CPLEX [2] solver, the model found the optimal solution in less than one second. Table 4 shows the placement of guests at tables. All the bride's guests were placed at table 1 and all the groom's guests were placed at Table 2.

As another scenario, suppose there are five tables available and each table seats four guests (m = 17, n = 5, a = 4, b = 1). The number of possible combinations is $4^{17} = 7.63 \times 10^{11}$. In this case, the model found the optimal solution in less than 2 seconds. Table 4 shows the guest placement. Again, there is a perfect partition between bride's guests and groom's guests and each guest knows at least one other person at their table. In addition, Abby is seated at Martha’s table, which was encouraged by giving them a connection of 10.

j	**Guest**	**Relation**
1	Deb	mother of the bride
2	John	father of the bride
3	Martha	sister of the bride
4	Travis	boyfriend of Martha
5	Allan	grandfather of the bride
6	Lois	wife of Allan
7	Jayne	aunt of the bride
8	Brad	uncle of the bride
9	Abby	cousin of the bride
10	Mary Helen	mother of the groom
11	Lee	father of the groom
12	Annika	sister of the groom
13	Carl	brother of the groom
14	Colin	brother of the groom
15	Shirley	grandmother of the groom
16	DeAnn	aunt of the groom
17	Lori	aunt of the groom

Table 2: Guest List

j/*k*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	1	50	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
2	50	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
3	1	1	1	50	1	1	1	1	10	0	0	0	0	0	0	0	0
4	1	1	50	1	1	1	1	1	1	0	0	0	0	0	0	0	0
5	1	1	1	1	1	50	1	1	1	0	0	0	0	0	0	0	0
6	1	1	1	1	50	1	1	1	1	0	0	0	0	0	0	0	0
7	1	1	1	1	1	1	1	50	1	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	50	1	1	0	0	0	0	0	0	0	0
9	1	1	10	1	1	1	1	1	1	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	1	50	1	1	1	1	1	1
11	0	0	0	0	0	0	0	0	0	50	1	1	1	1	1	1	1
12	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
13	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
14	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
15	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
16	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
17	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Table 3: Connection Matrix, C^{jk}

Guest	**Table #**	**Table #**
	(n = 5, a = 4, b = 1)	**(n = 2, a = 10, b = 1)**
Martha	1	1
Travis		
Abby		
Jayne	2	
Brad		
Deb	3	
John		
Lois		
Allan		
Annika	4	2
Carl		
Colin		
Lori		
Lee	5	
Mary Helen		
DeAnn		
Shirley		

Table 4: Example Problem Results

Full Problem

For the full problem, $m = 107$, $n = 11$, $a = 10$, and $b = 0$. There are a total of $11^{107} = 2.69 \times 10^{111}$ combinations for seating guests at the various tables, making this a much larger problem to solve. The model ran for 36 hours on one node of a Beowulf cluster with two 2.83 GHz Intel E5440 quad core processors and 16GB of memory. The best solution found after 36 hours gave a very good starting point for seating assignments, and only required a few rearrangements to please the mother of the bride.

Discussion

The crux of the model is the connection matrix, which is constructed to give great flexibility to users in reporting the relationships between guests. It can be as simple as giving a value of 1 if two guests know each other, and 0 otherwise. In our experience, this does not aptly capture the intuition one would apply when seating guests by hand. Thus, the matrix can be constructed using various weights, which depict how well two guests know each other.

We found that the most important assignment was to place guests who arrived as dates to the wedding at the same table. This was done by giving those guests a very high weight (50) in the connection matrix. From there, one could construct a level of weights for guests of the same immediate family, same extended family, etc. If one already has an idea of which guests should sit at the same table, but needs help filling out the tables, weights can be assigned based upon guests who should sit together. This is the method we imposed in our problems. Dates were given a weight of 50, guests who should sit at the same table a weight of 10, guests who know each other but do not necessarily need to sit at the same table a weight of 1, and guests who do not know each other a weight of 0. Additionally, negative weights could be assigned to discourage the model from seating two people at the same table (for example, divorced parents). The model successfully seats guests at appropriate tables, but only if the connection matrix is constructed properly. This mathematical representation is applicable to many different kinds of events, beyond weddings, such as bar/bat mitzvahs, retirement parties, anniversary parties, gala events - any event that requires a seating chart.

Acknowledgments

MLB gratefully acknowledges JDLP for still agreeing to marry her after writing this.

References

[1] Booke A and Kendrick D and Meeraus A (2007) General Algebraic Modeling Language (GAMS), Version 22.6. http://www.gams.com.
[2] IBM ILOG CPLEX, Version 12.1. http://www.ilog.com/products/cplex.