When I got married in 2013 one of the practical problems that my wife and me had to solve was creating a seating plan for the dinner party. This can be a both time consuming and not necessarily very fun activity, and not something we were looking forward to. It however turned out to be a good opportunity for testing OR methods on a very practical problem. This article explains how the problem of creating a plan for the seating arrangement at our wedding was handled. We see both how the problem can be formulated as mixed integer linear program and what data was used in the model. We also look at what corners had to be cut to solve the model, and finally we evaluate if the proposed seating plan was actually useful.

What is a good seating plan?

If we want to model the problem as an optimization problem the first step is to decide what we believe is a good seating plan. Afterwards we can then try to formalize the ideas. The following are the characteristics of a good seating plan that my wife and I identified, and are of course quite subjective.

Of the more trivial requirements we have that each guest should have exactly one seat, and that each table has neither too few guests nor too many (at our wedding each table could seat up to eight or nine guests). Besides from these trivial requirements we want each guest to be seated with people they have things in common with, and we also want each person know some of other the other guests at the table, but not necessarily all of them. Furthermore we want couples to be seated together, and as we have a bad habit of trying to play the matchmaker between our friends we also include what we believe would make great couples. In the same way some people do not necessarily enjoy each other's company, so we can consider adding such a rule. Finally we also want an approximately even number of men and women at each table.

All of these considerations are not very exact, and our goal is therefore not to get the best or optimal solution to a formalized model, but just some solution that is good enough.

Modeling the seating plan as an optimization problem

To model all of the requirements from above we need some information about the guests. We made use of the following data:

1. A guest list: Contains information about each guest, such as name, gender, age and a list of interests.
2. A list of couples: Contains the names of people who are a couple or that we would like to be a couple. More formally the tuple $(i, j) \in P$ if guest i and guest j are a couple.
3. A list of who knows whom: For each guest we would like to know which other guests does that person know. Specifying this for each guest is however very cumbersome. Instead we specify the relationships in groups of people where everybody knows each other. This is a quite efficient way of storing the relationships, as they there are very often large cliques of people where everybody knows everybody else. We can then process the input, and get a matrix of which guests knows which

other guests. Formally the parameter R_{ij} is 1 if guest i and guest j knows each other.

Entering the needed data is a time consuming process, and preferably a more advanced version of the software can draw the data from social networks.

An overview of the variables we need to model this as a linear program can be seen in Figure 1. The variable s_{it} is 1 if guest i is seated at table t, and where I is the set of all guests and T is the set of all tables. Variables m_t and f_t models the surplus of men (women resp.) at the table t. We describe how we use these two sets of variables in more detail when we introduce the constraint in which they appear. Then if the two guests i and j are seated at the same table t then the variable t_{ijt} should be 1 indicating that they are seated together. The last variable k_i is a slack variable that is positive when guest i knows less than three other people at the table he or she is seated at. The only variables we have to choose values for are the s_{it} variables, as the rest are then dependent on these decisions.

$$
\begin{aligned}
& s_{it} \in \{0,1\}, && \forall i \in I, t \in T \\
& m_t \geq 0, && \forall t \in T \\
& f_t \geq 0, && \forall t \in T \\
& t_{ijt} \in \{0,1\}, && \forall (i,j) \in I \times I, t \in T \\
& k_i \geq 0, && \forall i \in I
\end{aligned}
$$

Figure 1: Variables used in the model

Of the requirements mentioned above we model the following as hard constraints that must be observed in the seating plan:

- There is a capacity limit of nine guests at each table
- Each guest has exactly one seat at one table
- Couples must sit together

The rest of the requirements are modeled by taking them into account in the objective function with different weights. If we have a minimization problem the contributions are:

- Age difference between two guests seated at the same table, with weight 1/10 per year difference
- Gender imbalance at a table, with weight 1
- Guests knowing too few people at the table, with weight 10
- Guests with similar interests sits at the same table, with weight -1 per shared interest

$$\sum_{t \in T} s_{it} = 1, \qquad \forall i \in I \qquad (1)$$

$$\sum_{i \in I} s_{it} \leq 9, \qquad \forall t \in T \qquad (2)$$

$$s_{it} - s_{jt} = 0, \qquad \forall (i,j) \in P, t \in T \qquad (3)$$

$$\left(\sum_{i \in I} M_i s_{it} - F_i s_{it}\right) - m_t \leq 2, \qquad \forall t \in T \qquad (4)$$

$$\left(\sum_{i \in I} F_i s_{it} - M_i s_{it}\right) - f_t \leq 2, \qquad \forall t \in T \qquad (5)$$

$$t_{ijt} - s_{it} \leq 0, \qquad \forall i \in I, j \in I, t \in T \qquad (6)$$

$$t_{ijt} - s_{jt} \leq 0, \qquad \forall i \in I, j \in I, t \in T \qquad (7)$$

$$t_{ijt} - s_{it} - s_{jt} \geq -1, \qquad \forall i \in I, j \in I, t \in T \qquad (8)$$

$$k_i - 3s_{it} + \sum_{j \in I : i \neq j} R_{ij} t_{ijt} \geq 0, \qquad \forall i \in I, t \in T \qquad (9)$$

Figure 2: Constraints include in the model

All the constraints we include can be seen in Figure 2. Constraints (1) ensure that each guest has been assigned to exactly one table. The constraints (2) model that every table can have at maximum nine guests. Constraints (3) ensure that every couple is seated together. We use constraints (4) and (5) to model that we would like almost the same number of men and women at every table. The idea in the constraint (4) is that if there are two men more than women the variable m_t has to be positive which incurs a penalty in the objective function. The constraints (5) model the same just for more women than men at the table. The constraints (6)-(8) force the variables t_{ijt} to be 1 if the two guests i and j both are seated at table t and 0 otherwise. The variable is used in the objective function to penalize any age difference between the two guests and reward any shared interests between them. The variables are also used in constraints (9) that forces k_i to be positive if a guest does not know at least 3 other guests at the table they are seated.

The objective function that we want to minimize is:

$$\sum_{i \in I} \sum_{j \in I} \sum_{t \in T} C_{ij} t_{ijt} + \sum_{t \in T} (f_t + m_t) + 10 \sum_{i \in I} k_i$$

The coefficient C_{ij} is an aggregate value that takes into account both the age difference and shared interests between the two guests i and j.

Solving the optimization problem

To solve the model we use the SCIP Optimization Suite. It can however not solve the full model with 82 guests immediately, so we have to make a couple of adjustments. First, it is quite natural that our parents and our two remaining grandparents have to be seated with us. This match with the approximately eight people that can sit at a table, and means that one table can be removed from the optimization problem.

When solving smaller instances it can be seen that the model allows a lot of symmetric solutions. The symmetry comes from swapping all the guests sitting at one table with all the guests sitting at another table. This gives a symmetric solution with exactly the same cost; the only difference is the labeling of the tables. To remove this symmetry we choose a number of guests, whom we do not think, should be seated together and assign them to different tables. This limits the solution space and will remove the optimal solution if we made a mistake in choosing the fixed guests. But as mentioned above we are looking for a good solution, not necessarily the optimal one. The modification breaks the symmetry since now the fixed guests cannot be swapped between tables. There are more elegant ways to handle this, but the shortcut of assigning one guest to each of the tables was chosen for ease of implementation. With these two adjustments we can solve the full problem with SCIP in around 10 hours.

Could the solution be used?

After letting the SCIP Optimization Suite run overnight we looked at the first automatically generated seating plan, and were a little disappointed. The proposed plan was not very good, but after a little inspection it was clear that we had not entered enough data about the guests. So we did a couple of iterations of inspecting the solution, entered more data on the guests and re-solved the model – especially the list of interests for each guest had to be update quite a bit. This improved the quality of the seating plan significantly and the final plan was used directly, with the small modification that we swapped two guests and moved one.

It is difficult to objectively evaluate the quality of a seating plan, but our guess is that it would have been a non-trivial task to create a plan of similar quality by hand. So while we did not save any time by doing this it was feasible and a lot more fun. The resulting software is also reusable, and two of our friends, who are getting married soon, are trying to use the same model for planning the seating arrangements at their wedding. On a final note, our efforts to match make our friends were in vain, as we did not manage to setup any new couples.

Author biography:
Niels Kjeldsen is a software engineer with Schneider Electric where he helps develop a software suite for running datacenters efficiently. He has an Industrial PhD, which was done in cooperation with DONG Energy and the University of Southern Denmark. He is interested in complex optimization problems (especially within the energy sector) and in software development practices.