Chapter 6
Designing Seating Plans

Chapters 6, 7 and 8 each contain a detailed case study showing how graph colouring methods can be used to successfully tackle important real-world problems. The first of these case studies concerns the task of designing table plans for large parties, which, as we will see, combines elements of the NP-hard (edge) weighted graph colouring problem, the equitable graph colouring problem and the $k$-partition problem. A user-friendly implementation of the algorithm proposed in this section can also be found online at www.weddingseatplanner.com.

6.1 Problem Background

Consider a social event such as a wedding where, as part of the formalities of the day, $N$ guests need to be divided among $k$ dining tables. To ensure that guests have a good time it will often be necessary to design a seating plan, thereby allowing guests to be seated at tables with appropriate company. The following sorts of factors might be taken into account:

- Guests belonging to groups, such as couples and families with small children, should be seated at the same tables, preferably next to each other.
- If there is any perceived animosity between different guests, they should be seated at different tables. Similarly, if guests are known to enjoy one another’s company, it may be desirable for them to be seated at the same table.
- Some guests might be required to sit at a particular table (e.g. close to the kitchen or washrooms). Similarly, some guests might be prohibited from sitting at certain tables.
- Since tables could vary in size and shape, each table should be allocated a suitable number of guests, and these guests should be arranged around the table in an appropriate manner.

A naïve method for producing a seating plan best fitting these sorts of criteria might be to consider all possible plans and then choose the one perceived to be the most...
suitable. However, for nontrivial values of $N$ or $k$, the number of possible solutions is too large for this to be possible. To illustrate, consider a simple example where we have $N = 48$ guests using $k = 6$ equal sized tables (i.e., exactly eight guests per table). For the first table we need to choose eight people from the 48, for which there are $\binom{48}{8} = 377,348,994$ possible choices. For the next table, we then need to choose eight further guests from the remaining 40, giving $\binom{40}{8} = 76,904,685$ further choices, and so on. Assuming that $k$ is a divisor of $N$, using equal sized tables, the number of possible plans is thus:

$$\prod_{i=0}^{k-2} \binom{N-i(N/k)}{N/k}.$$  \hfill (6.1)

This function clearly has a growth rate that is subject to a combinatorial explosion—even for the modestly sized example above, the number of distinct solutions to check is $\prod_{i=0}^{k-2} \binom{N-i(N/k)}{N/k} \approx 3.8 \times 10^{33}$ which is far beyond the capabilities of any state-of-the-art computing equipment. Furthermore, if we were to relax the problem by allowing $k$ tables of any size $\geq 1$, the task would now be to partition the $N$ guests into $k$ nonempty subsets, meaning that the number of solutions would be equal to a Stirling number of the second kind (Equation (1.6)). These numbers feature even higher growth rates than Equation (6.1), giving even larger solution spaces.

Such features demonstrate that this sort of naïve method for producing a desirable seating plan is clearly infeasible in most cases. However, the problem of constructing seating plans is certainly important since (a) it is regularly encountered by party organisers and (b) the quality of the proposed solution could have a significant effect on the success (or failure) of the gathering.

Currently there is a small amount of commercial software available for constructing seating plans, such as Perfect Table Plan, Top Table Planner, and Seating Arrangement.\footnote{Refer to the websites www.perfecttableplanner.com, www.toptableplanner.com, and www.seatingarrangement.com respectively.} The first of these examples allows users to input a list of guest names into the system and then specify preferences between these guests (such as whether they need to be seated apart or together). It then allows users to define table shapes, sizes and locations, before assisting the user in placing the guests at these tables via drag and drop functionality and also an auto assign tool. The exact details of the underlying algorithm used with the auto assign tool are not made public by the software vendor, though its online documentation states that an evolutionary algorithm is used, with different penalty costs being applied for different types of constraint violation. The fitness function of the algorithm is simply an aggregate of these penalties.