Planning the Seating Chart for Optimal Weddings
Project Plan

Daniel Joffe

u03dj17@abdn.ac.uk

Department of Computing Science,
University of Aberdeen, Aberdeen AB24 3UE, UK

Introduction

The wedding dinner seating chart is a classic optimisation problem: given a list of guests and knowledge of their relationship to one another, separate them into tables with the goal that everyone is seated with as many friends as possible and away from anyone they're unfriendly with. This task is typically done by hand, which can be a lengthy process for large weddings. The goal of this project is to find a way to automate the process.

Though framed in the language of weddings, the underlying computational problem has other applications. For example, a teacher may wish to find a composition of teams for a group project that will lead to minimal in-fighting.

Goal

My goal is to identify a heuristic that solves the seating chart problem well, and to make quantifiable statements about how well and how quickly. In the event that some heuristics sometimes perform better than others, I want to be able to say under which circumstances which heuristic performs best.

Methodology

- Read previous works relating to combinatorial optimisation problems.
- Formalise the problem - create a model of guest preferences and choose the metrics that will be used to compare seating charts.
- Implement the following heuristics: the integer programming approach used by Bellows and Peterson [1], a genetic algorithm, and hill-climbing.
- Generate a large number of larger problem instances.
- Test the implemented heuristics on the larger dataset.
- Compare the implemented heuristics by the metrics previously decided on.

- Write a report detailing the process, conclusions, and suggesting areas of further research; create a poster for summary.

Resources Required

I will require a development computer with a general-purpose programming language installed, such as Python or Rust. I will need access to integer programming software such as IBM's CPLEX or Google's OR-tools. For some testing, I may need access to a high performance computing service such as the University of Aberdeen's Maxwell.

Risk Assessment

- Implementing the heuristics correctly may take too long. If this is the case I will drop one and study the ones remaining.
- The project may take too *little* time. In this case, I can study a wider range of heuristics to cover more ground.
- Collecting data from large-scale testing may take too long due to inefficient heuristics. That in itself is already useful data, but there are steps I can take to minimise the impact on my study. I can run my code on a faster computer. I can test the heuristics on a range of input-sizes to determine where they become infeasible.

Timetable

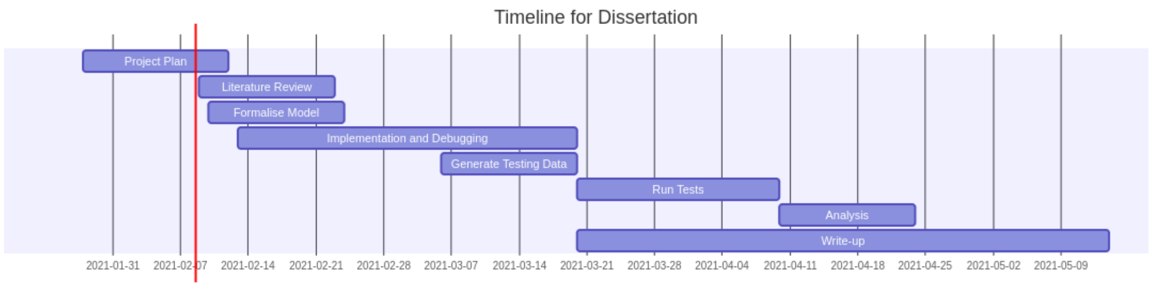

Figure 1: Main Project Activities

References

[1] Meghan L. Bellows and J. D. Luc Peterson. Finding an optimal seating chart for a wedding. *Improbable*, 2012.