MISG 2017 TV-Scheduler Report

Dr Nick Hale, Shane Josias

January 2017

Introduction

When determining a TV schedule for a particular channel, it might be intuitive that a certain combination of shows (movies) would perform better than some other combination. Furthermore, there should be constraints placed on which combinations are viable. This could include, for example, the idea that shows which have an age restriction cannot be played at inappropriate times. This then becomes an optimisation problem, where we maximise some measure of score while conforming to the constraints imposed. This document will highlight a Binary Integer Linear Programming approach to solving this optimisation problem. Elaboration will be given to the formulation of the cost function and constraints. Thereafter, the tools used in the solution of the problem, and results will be discussed.

1 Data

- **Movie Data** (duration, acceptable playing hours, min/max number of runs)
- **Time Data** (1-hour timeslots and their scores)
- **Channel-Specific Rules** (peak start/end, peak/offpeak gaps)

2 Constraints

- One movie at a time (physical constraint)
- Min/max number of runs
- Min movie repeat time (48 hours)
- Acceptable playing hours (e.g., age constraints)
- License start/stop
- 1 week ban
- Length of (advert) breaks

3 Model Formulation

We model the optimisation problem as a Binary Integer Linear programming problem. We first describe a Linear Programming problem.

A Linear Programming problem seeks to optimise a linear cost function with respect to certain linear equality or inequality constraints. An Integer Linear Programming problem requires that the solution variables are restricted to `integers`. When the solution vector is further restricted to either 0, or 1, we have a `Binary Integer Linear Programming problem`:

$$\begin{array}{ll} \text{Maximise} & \mathbf{c}^\top \mathbf{x} \\ \text{subject to} & \mathrm{A}\mathbf{x} \leq \mathbf{b} \\ \text{with} & \mathbf{x} \in \{0, 1\} \end{array}$$

There exist numerous efficient algortihms that can be used to solve such problems. The hope is that, if the scheduling problem can be reduced to a BILP, then we can solve it.

4 Assumption and Derivation

We first turn our attention to a simpler problem by making a significant assumption. We assume that all movies are 2-hours long. Perhaps this is not unreasonable when we consider that the average length of a movie is 1.77 hours.

We proceed with our two-hour assumption and divide each day into 12 2-hour long blocks, each with an associated score. The data set for channel 1 has 31 days (making 372 slots to fill) and 76 movies. Now consider a 76×372 (binary) state matrix X whose (i, j) entry is 1 if movie i is played at time j. This state matrix is then collapsed to a 28272×1 state vector $\mathbf{x}$. Similarly we make a score matrix C whose (i, j) entry is the product of the score associated with movie i and time-slot j (given by the data) and collapse this to a vector $\mathbf{c}$.

Our aim is to find the binary vector $\mathbf{x}$ which maximises $\mathbf{c}^\top \mathbf{x}$. Importantly, the constraints need to be formulated such that they can be represented as a linear system.

5 Constraints

Most of our constraints are 'counting' constraints, which can be easily implemented as linear (in)equalities:

One movie at a time:

$$\left(\begin{array}{cccc|cccc|cccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right) \mathbf{x} = \left(\begin{array}{c} 1 \\ 1 \\ 1 \end{array}\right)$$

Max number of movies repeats:

$$\left(\begin{array}{cccc|cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right) \mathbf{x} \leq \left(\begin{array}{c} 1 \\ 12 \\ 0 \\ 3 \end{array}\right)$$

(For min number of repeats change $\leq$ to $\geq$.)

No repeats within 48 hours: Consider for now just the constraint that we can't play movie #1 twice in a row. The idea for 48 hours is similar.

$$\left(\begin{array}{cccc|cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{array}\right) \mathbf{x} \leq \left(\begin{array}{c} 1 \\ 1 \end{array}\right)$$

(For min number of repeats change $\leq$ to $\geq$.)

24 hours in a day:

$$\left(\begin{array}{cccc|cccc|c} 92 & 75 & 120 & 115 & 92 & 75 & 120 & 115 & 0 \quad \dots \end{array}\right) \mathbf{x} \leq 24$$

Earliest Allowed: Since each partition of the coefficient vector corresponds to a time slot, excluding movie m_i from time slot t_j is equivalent to removing the column of that movie in the coefficient matrix. The corresponding entry in the state vector $\mathbf{x}$ must also be removed. Suppose we must have that movie m_3 cannot be played in timeslot t_1, then in the one movie at a time constraint, the matrix becomes:

$$\left(\begin{array}{cccc|cccc|cccc} 1 & 1 & \mathbf{1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{0} & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{array}\right) \left(\begin{array}{c} x_{11} \\ x_{21} \\ \mathbf{x_{31}} \\ x_{41} \\ x_{12} \\ \vdots \end{array}\right) = \left(\begin{array}{c} 1 \\ 1 \\ 1 \\ \vdots \end{array}\right)$$

6 Tools

The algorithms available for BILPs come in different flavours. An initial implementation was done in MATLAB, but due large computational times we switched to the propriety Gurobi software (www.gurobi.com - free for academic use!).
More specifically, available software includes:

- `intlinprog` in MATLAB.
- `scipy.optimize.linprog` in Python.
- Gurobi (MATLAB and Python interfaces)
- LINGO/LINDO
- MPL/CPLEX

7 Results

7.1 2-Hour assumption

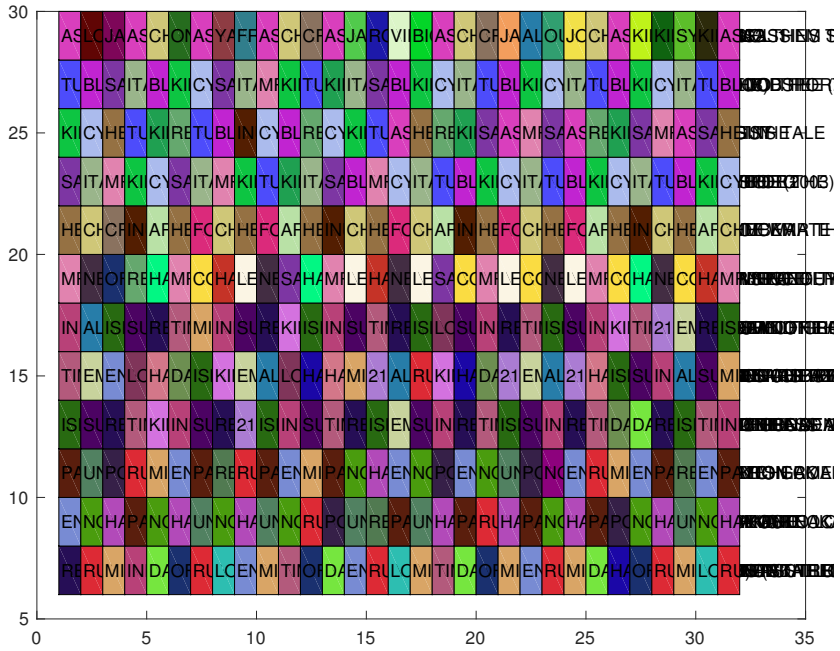

Figure 1: Visualisation of Solution

Here we see that the movies are divided into 2 hour slots. At present, while there are gaps between movies, it is not clear. One would have to do a secondary

run to determine where empty slots are present and possibly fill those without violating the constraints.

7.2 2-Hour Assumption removed

To remove the 2-hour assumption we could split the timeslots up into smaller windows (e.g. 15min windows) and add some form of linear continuity constraint. While the idea is attractive, at the time of implementation, a solution to adding the linear continuity constraint could not be found. Additionally, one might attempt to allow variable slot-sizes, but this leads to a non-linear cost function.

To solve the problem as best we could, we attempted the following:

- Treat two hour assumption a model for the real problem
- Add a $\leq$ 24hr constraint on the duration for each day (linear!)
- Solve model (discrete time) problem
- Convert to continuous time
- Apply some post-processing / further optimisation
- Compute score using provided R code

The 24hr constraint ensures that shows will not run into the next day. There may be gaps between the shows, and the post-processing step could address this. Its clear from the Figure 2 that there are gaps at the beginning and end of each day, which will need to be filled. The schedule was then tested with `schedTester.R`.

Project	Existing	BILP
1	15.6	17.6
2	3.9	5.5
4	3.9	4.9
6	3.7	4.8

Table 1: Table of Results

Although the BILP performed better there were some constraints violated. The constraints violated include license start/end dates (which was not catered for) and in some cases earliest allowed too. The earliest allowed reported a violation when an *unsavory* film was played at the early hours of the morning. One could really argue whether this is really a violation.

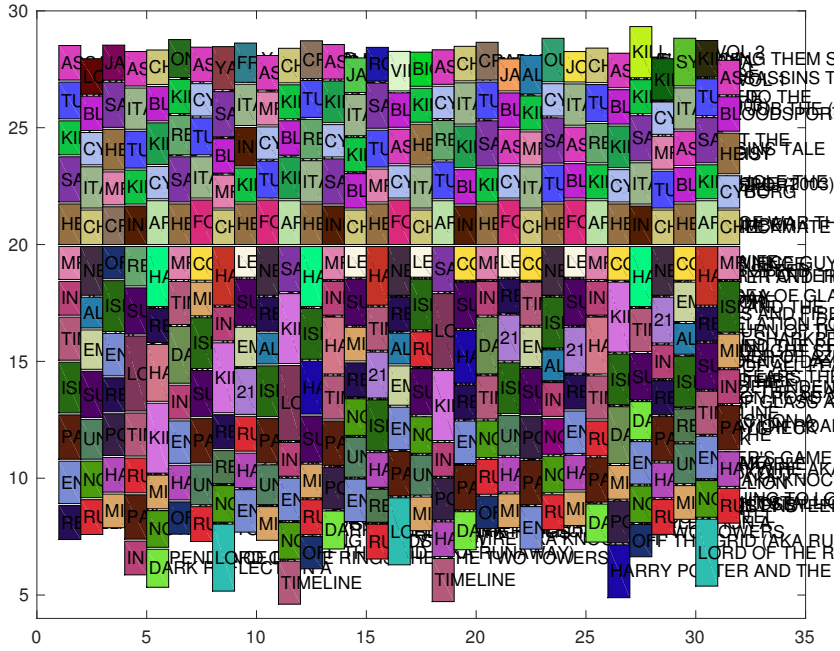

Figure 2: Visualisation of Solution

8 Future Work

Since the 2-hour assumption is a pretty big one to make, it might be a good idea to improve on this. Dr Hale proposed a solution in which the 2 hour constraint could be removed more effectively. He mentioned that the idea of splitting the time-slots into smaller intervals of say 15minutes would be a good approach. However, what was missed before was that in order to impose a continuity constraint , we would need to split the movies into smaller intervals as well. In this way we will be able to ensure that a movie's sub-parts would follow on after one another. With this approach, the gaps between shows constraint is taken care of more effectively. The previously mentioned constraints would have to change somewhat to incorporate this approach.
It might also be a great addition to create the system such that results can be tweaked. It might be that the system spits out a solution which is considered optimal according to the constraints, but not the most desired. Adding some functionality to allow subtle human evaluation might improve the schedule's effectiveness. Ultimately, a second processing process, or optimisation process would need to be run in order to achieve the most desired outcome. This is, of course, due to the fact that not all constraints can effectively be included. The second-optimisation run could ensure that there are valid gaps between shows. The constraints themselves could be researched and possibly improved on. Take for example the 48-hour constraint: We could play the highest rated

movie repeatedly with a 48-hour gap in between, but it might not be desirable. The second optimisation step could then attempt to improve the *viewership* rating of the schedule, by tweaking the schedule such that certain situations are avoided. Another improvement could include taking a more global viewpoint: instead of viewing the channels as independant of one another, consider the case where channels could be competing with one another. As an example, a live sports match on a sports channel could compete with a high-rated movie in the same time-slot. While this increases the size of the problem tremendously, it might be interesting to find out if this bears interesting results.