Movie Scheduling Problem

Cindy Li **Lincole Li** **Jerick Shi** **Maolinsheng Ye**

Abstract

Movie theaters face a persistent challenge in scheduling films due to the distinct release intervals of specific movies. This task involves decisions such as selecting featured movies, allocating screens, determining screening times, and setting exhibition durations. Current scheduling procedures, primarily reliant on manual processes by "movie programmers," lack efficiency and often fail to generate optimal solutions, compounded by time constraints. To address this issue, our paper proposes reframing the movie scheduling challenge as a linear programming problem, which finds an optimal solution that maximizes profitability by assigning movie screening times. We then apply our model on a case study and then a larger simulated dataset. Using our model, we observe several patterns regarding effects of different parameters and answer the question of why movie theatres rent movies more often than buying them.

1 Introduction

The scheduling of movies has posed a persistent challenge for various movie theaters. In light of the fact that specific movies are released at distinct intervals, movie theaters are confronted with a multitude of decisions. These decisions encompass the selection of movies to be featured, the allocation of screens or rooms for each movie, the determination of screening times, and the duration for which a particular movie should be exhibited, spanning across a specified number of weeks.

Nevertheless, in practice, the scheduling of movies predominantly relies on manual adherence to specific procedures by the designated "movie programmers," who, largely based on their intuition, strive to optimize the schedule for profitability without inducing temporal or spatial conflicts. Furthermore, the allotted time for the scheduling task is typically constrained. In summation, the existing scheduling procedures lack efficiency and do not ensure the generation of optimal solutions.

In this paper, we endeavor to formulate the movie scheduling predicament as a linear programming challenge, wherein the optimal solution, denoting the selection of movie schedules, yields the maximum profit for the movie theatre. By employing this mathematical framework, we aim to enhance the efficiency of the scheduling process and derive superior solutions. Addressing this objective necessitates the anticipation of revenue and cost implications associated with scheduling a specific movie in a designated room at a specified time. Subsequently, we endeavor to ascertain the schedule that maximizes projected profits while adhering to various constraints.

After establishing the model, we then experiment on it by performing sensitivity analysis with the different parameters, and then answer the hypothesis on whether it is worth more for a movie theatre to rent or purchase a movie.

1.1 Background/Literature

The motion picture industry is becoming increasingly focused on by market researchers and academics. There are numerous problems which are encapsulated in the consumption of movies, including the forecasting of popularity, release timing, effects of advertisements, as well as our main focus in this project– movie scheduling. Traditionally, intuition or loose rules without rigid structures have been utilized to make decisions related to scheduling movies, while there is significant room for improving theatre efficiency and profitability with the implementation of optimization algorithms.

There is an extensive list of variables and factors which affect the decision-making process in multiplexes. Existing research and case studies often propose models based on varying assumptions. In this section we will provide an overview of the exising research in this realm of optimization applications, specifically the approaches used by studies and results derived.

In previously established studies and formulations of solutions to the movie-scheduling problem, solutions have taken the form of a directed graph, with nodes representing movies being played in a specified theatre at a specified time (Eliash et al. (2)), and, more commonly, as linear integer programs with varying assumptions, using decision variables to encode the screening of a specified movie on a specified theatre screen at a specified time, with constraints and an objective function, which is usually to maximize a profit or projected box office revenue (Edet et al., Swami et al. (1)). Constraints typically ensure that one screen is only playing one movie at a given time, and that rental and theatre capacities respected.

2 Methods/Model

2.1 Introduction of Variables

We formulate the movie scheduling problem as an integer program. Our goal is to schedule the movies to the different screens during different time periods such that we maximize profit. Suppose that there are M different movies $\{m_1, \ldots, m_M\}$ and N time periods $\{t_1, \ldots, t_N\}$. Each time period can either represent a day, an hour, or a week, depending on the specification. In addition, we assume that the movie theatre has S different screens $\{s_1, \ldots, s_S\}$.

We first define the following parameters:

- p_j: The capacity of the room with screen s_j
- b_i: The beginning time period in which movie m_i starts playing
- $r_{i,j,k}$: the revenue generated by playing movie m_i on screen s_j on the k^{th} time period after m_i's first screening
- p: the proportion of the total screens that any single movie can take up during any time period
- POP_i: the popularity of movie m_i
- RC_i: the rental cost of movie m_i for a single time period

- OC: constant operation of cost of movie theater for any time period

We then define the decision variable $x_{i,j,k}$ and $y_{i,k}$ as two binary variables such that

$$x_{i,j,k} := \begin{cases} 1, & \text{if } m_i \text{ is played on } s_j \text{ during the } k^{\text{th}} \text{ time period after its first screening} \\ 0, & \text{otherwise.} \end{cases}$$

$$y_{i,k} := \begin{cases} 1, & \text{if } x_{i,j,k} = 1 \text{ for at least one } k \\ 0, & \text{otherwise.} \end{cases}$$

2.2 Assumptions

(a) We rent each movie right right after it comes out. In other words, each movie is associated with a rental start time. We also assume there is no time lag between a movie's order placement and its arrival. Note this assumption is grounded in practicality, as the initial release period typically represents the peak popularity of a movie.

(b) The logistical expenses associated with projecting a movie on a specific screen can be approximated by considering the dimensions of the screen or room.

(c) The anticipation and public appeal of a particular movie can be forecasted and evaluated through the application of machine learning techniques, as expounded upon in the detailed illustration provided in Section 3.

(d) The operational costs per unit of time remain consistent across all time periods. The validity of this assumption hinges upon the presumption of a consistent and stable operational cost structure within movie theatres, regardless of temporal considerations such as the day of the week, month, season, or the specific movies being exhibited.

2.3 Constraints

We will initiate the formulation of constraints by establishing the decision variable as $x_{i,j,k}$. A fundamental constraint is evident, stating that two movies cannot be concurrently scheduled for screening on the same screen. Notice that this only happens if on the same screen, for 2 movies m_p and m_q, $b_p + k_p = b_q + k_q$. We then write the formulation of the constraint:

$$\sum_{i \in [M]} x_{i,j,t-b_i} \leq 1, \qquad \forall j \in [S], t \in [N]$$

Furthermore, in order to ensure that each movie is played for a continuous set of time periods (or not at all), we need to add a new constraint. We introduce a new binary variable $y_{i,k}$, which is equal to 1 if $x_{i,j,k} = 1$ for at least one j and 0 otherwise. Intuitively, it indicates whether movie i is played at all during the k^{th} week after its first showing. This then gives us the following constraints:

$$y_{i,k} \leq y_{i,k-1} \qquad \forall k \in [N] \tag{1}$$

$$\sum_{j \in [S]} x_{i,j,k} \leq S \cdot y_{i,k}, \qquad \forall i \in [M], k \in [N] \tag{2}$$

Constraint (1) enforces that (a) if movie m_i is played in time period $b_i + k$, it must have been played in all the previous time periods, and (b) if a movie m_i is not played in time period $b_i + k$, then it can't be played in subsequent time periods. Constraint (2) enforces that $y_{i,k} = 1$ when $x_{i,j,k} = 1$ for at least one j, since if $y_{i,k} = 0$ in this case, the constraint won't hold. We will show later that in the

other case where $\sum_{j\in[S]} x_{i,j,k} = 0$, that will force $y_{i,k}$ to be 0 to maximize the *objective function* in the optimal solution.

Finally, in order to ensure that there is not a single movie that takes up all the screens at any given time period, we add a diversity constraint as follows:

$$\sum_{j\in[S]} x_{i,j,k} \leq \lfloor p \cdot S \rfloor \qquad \forall i \in [M], k \in [N] \tag{3}$$

Constraint (3) ensures that at any time period, one movie can take up at most a proportion of $p \in [0, 1]$ of the all screens in the movie theatre.

2.4 Objective Function

Our objective is to maximize total profit (TP), which we calculate to be the difference between the total revenue (TR) and total cost (TC):

$$\text{TP} = \text{TR} - \text{TC}.$$

2.4.1 Total Revenue

Notice that the total revenue will just be the amount of revenue produced by each movie by each screen at a certain time period all summed together, giving us the expression

$$\text{TR} = \sum_{i,j,k} x_{i,j,k} r_{i,j,k}.$$

As mentioned before, $r_{i,j,k}$ represents the predicted revenue of playing movie m_i on screen s_j at the k^{th} time period after its first screen, which isn't public information. Hence, we shall create a model to forecast its value.

We posit that the variable $r_{i,j,k}$ exhibits a *positive* correlation with both the popularity of the movie and the capacity of the venue in which it is exhibited. This claim is grounded in the anticipation that a show's increased popularity and larger screening capacities will result in a larger audience turnout.

In particular, there should be a linear relation between popularity and revenue since more people buying tickets corresponds to more revenue. On the other hand, we assume a log relation between room size and revenue, since doubling the room size shouldn't have as much as an effect on total revenue.

Furthermore, we also assume there is a *negative* relationship between $r_{i,j,k}$ and k, since the demand for a movie declines with time. This gives us our revenue model:

$$r_{i,j,k} = c \cdot \frac{\text{POP}_i \cdot \log(p_j)}{k}$$

where $c \in \mathbb{R}^+$ is the overall coefficient of the linear relationships, and POP_i represents the popularity of movie i. Typically, the popularity of a movie isn't given until weeks after it airs, so we provide a machine learning approach to predict the popularity of a model in section 3.

2.4.2 Total Cost

We assume that the cost in this model consists of renting costs and operation expenses. Furthermore, we assume that there is a fixed cost per time period for renting each movie, and there is a fixed cost

per time period for operating the theatre. Let RC_i be the rental cost of movie i for a single time period, and OC be the operation cost for a single time period. This then gives us the following expression:

$$\text{TC} = \sum_{i \in [M]} \sum_{k \in [N]} \text{RC}_i \cdot y_{i,k} + N \cdot \text{OC}. \tag{4}$$

Note that our requirement for renting movie m_i during the k^{th} time period post its initial screening is contingent upon the condition that m_i is presented at least once within that particular timeframe. Concurrently, we make the assumption that the theater operations will span the entirety of the N designated time periods.

To maximize the profit, we want to minimize TC. Thus, $y_{i,k}$ will be set to 0 whenever possible in the optimal solution. This explains why the definitions of $x_{i,j,k}$ and $y_{i,k}$ are consistent in constraints (1) and (2).

2.4.3 Model Formulation

Combining them, we have our final linear program to be:

$$\text{maximize} \qquad \sum_{i,j,k} x_{i,j,k} r_{i,j,k} - \left(\sum_{i \in [M]} \sum_{k \in [N]} \text{RC}_i \cdot y_{i,k} + N \cdot \text{OC} \right)$$

subject to

$$\begin{aligned}
\sum_{i \in [M]} x_{i,j,k-b_i} &\leq 1, && \forall j \in [S], t \in [N] \\
\sum_{j \in [S]} x_{i,j,k} &\leq S \cdot y_{i,k}, && \forall i \in [M], k \in [N] \\
\sum_{j \in [S]} x_{i,j,k} &\leq \lfloor p \cdot S \rfloor && \forall i \in [M], k \in [N] \\
x_{i,j,k} &\in \{0, 1\} \\
y_{i,k} &\in \{0, 1\}
\end{aligned}$$

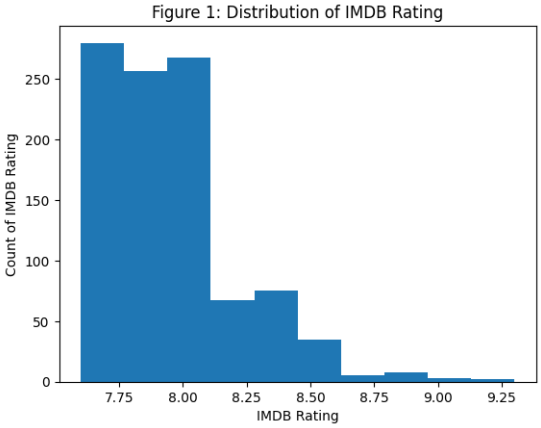
3 Popularity Prediction Model

As mentioned before, we need to build some model in order to predict the popularity of a movie. If a movie is relatively new, there might not be much information about it. Hence, we shall introduce an idea of using machine learning methods in order to predict the popularity of a movie. Our main goal here is to provide an approach, since there are several online resources that produce much optimal results.

3.1 Exploratory Data Analysis

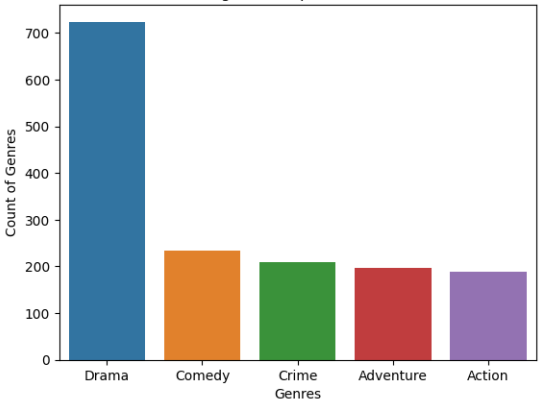
For our prediction, we shall use the IMDB dataset of the top 1000 movies and TV shows. We are particularly interested in whether we could use the genre of the movie, the director's name, the names of the leading actors/actresses, as well as the movie description in order to predict the final IMDB rating on a scale from 1 to 10.

Figure 1: Distribution of IMDB Rating



As seen in figure 1 above, most of the ratings are above 7 and below 9.5, with a significant skew to the right, which makes sense, since the movies are the top 1000 rated.

Figure 2: Top 5 Genres

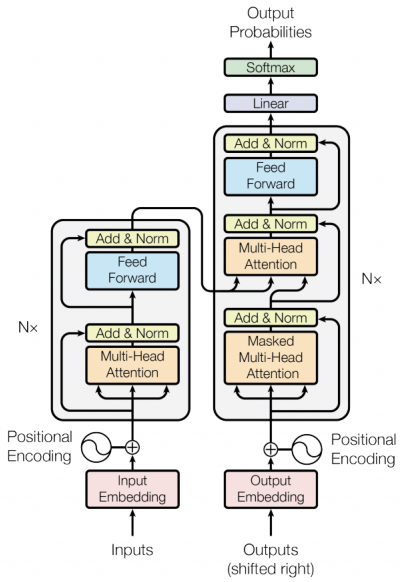


In figure 2, we see that most of the genres are drama, while the other common ones include comedy, crime, adventure, and action.

Hence, we have a diverse dataset in which we could train our model on.

3.2 Methods

Our predictor variables are mostly categorical. In particular, we shall one-hot encode the genre, director, as well as the different stars in the movies. For the summary, we shall Huggingface's transformer embedding model in order to embed the long summaries into a vector of different numbers (3):



Using the data, we then utilize 3 different methods to fit the data: Linear Regression, Neural Network, as well as Support Vector Machines. Details of models are shown in the appendix.

3.3 Results

We summarize the results of the different models in the table below:

R-Squared of All Models	
Model	R-Squared
Linear Regression	-0.1827
Neural Network	-3.1798
SVM	0.0748

Notice that most of the R-Squared values are negative, which indicates that the model performs worse than simply predicting the mean, which is expected, since we have thousands of different variables, a very simple baseline model, and values which are all quite close to each other. This could be greatly improved through further testing as well as using other possible deep learning methods.

4 Case Study

In this section we demonstrate a case study of the implementation of our program formulation, applied to a sample set of data which we construct ourselves.
For simplicity and applicability we first establish the following assumptions:

- Our theatre is open for 24 hours a day, 7 days a week;
- We start each movie on the hour, and each movie screening lasts exactly two hours (time needed for advertisements, preparation, cleaning up etc.;
- Each movie will be released at 0:00 on its release date, so on the beginning rental date of a movie, it will be available for screening at 0:00;
- The theatre will be filled for each screening that occurs (we don't apply the popularity measure in this case study);
- Our movie theatre consists of 6 screens total, 3 of which can hold 50 people, and 3 with a capacity of 100;
- Our total operational cost for the one-week time span over which we analyze this case study will be 20,000 dollars.

We construct our data set to consist of three movies: *Oppenheimer, Barbie*, and *Spirited Away*. Thus our set of movies is $\{m_1, m_2, m_3\}$, which correspond to each of the three movies we will screen during the week following the release of *Oppenheimer* and *Barbie*. We will have time 0:00 of July 21, the release date of the two movies, be t_1, and assume that *Spirited Away* is rented at this time period also.

Moreover, we will set the weekly rental prices for each of the movies as follows: $20,000$ dollars for *Oppenheimer*, $15,000$ dollars for *Barbie*, and $5,000$ dollars for *Spirited Away*. We will also set the movie ticket prices to be as follows: 30 dollars per ticket for *Oppenheimer*, 20 dollars per ticket for *Barbie*, and 10 dollars per ticket for *Spirited Away*.

Lastly, we set p to be $\frac{1}{2}$ in this model, thereby ensuring that no movie is allocated more than half of the available screens at any given time.

Under our assumptions and setups, and using the above data set as input, we utilized Gurobi to generate an optimal feasible movie schedule for the week following July 21.

We elucidate the movie schedule by presenting the generated timetable for screen 3 below. It is imperative to recognize that this schedule pertains to a singular screen and, therefore, does not depict the comprehensive movie schedule for the entire week.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day7
0:00-2:00	Barbie	Barbie	Oppenheimer	Oppenheimer	Spirited Away	Spirited Away	Spirited Away
2:00-4:00	Barbie	Barbie	Oppenheimer	Oppenheimer	Spirited Away	Spirited Away	Spirited Away
4:00-6:00	Barbie	Barbie	Spirited Away	Spirited Away	Spirited Away	Spirited Away	Spirited Away
6:00-8:00	Barbie	Barbie	Spirited Away	Spirited Away	Spirited Away	Spirited Away	Spirited Away
8:00-10:00	Barbie	Barbie	Spirited Away	Spirited Away	Spirited Away	Spirited Away	Spirited Away
10:00-12:00	Barbie	Barbie	Spirited Away	Spirited Away	Spirited Away	Spirited Away	Spirited Away
12:00-14:00	Barbie	Barbie	Spirited Away	Spirited Away	Spirited Away	Spirited Away	Spirited Away
14:00-16:00	Barbie	Barbie	Oppenheimer	Spirited Away	Spirited Away	Spirited Away	Spirited Away
16:00-18:00	Barbie	Barbie	Oppenheimer	Spirited Away	Spirited Away	Spirited Away	Spirited Away
18:00-20:00	Barbie	Barbie	Oppenheimer	Spirited Away	Spirited Away	Spirited Away	Spirited Away
20:00-22:00	Barbie	Barbie	Oppenheimer	Spirited Away	Spirited Away	Spirited Away	Spirited Away
22:00-24:00	Barbie	Oppenheimer	Oppenheimer	Spirited Away	Spirited Away	Spirited Away	Spirited Away

The objective value of the schedule generated for this specific case study was $185,967$ dollars in total.

Now to establish the effects of the "diversity constraint" which we included in our model formulation, we will show the Gurobi-generated solution upon removing this constraint.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
0:00-2:00	Oppenheimer	Oppenheimer	Spirited Away	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
2:00-4:00	Oppenheimer	Oppenheimer	Spirited Away	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
4:00-6:00	Oppenheimer	Oppenheimer	Spirited Away	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
6:00-8:00	Oppenheimer	Oppenheimer	Spirited Away	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
8:00-10:00	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
10:00-12:00	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
12:00-14:00	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
14:00-16:00	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
16:00-18:00	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
18:00-20:00	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
20:00-22:00	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer
22:00-24:00	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer	Oppenheimer

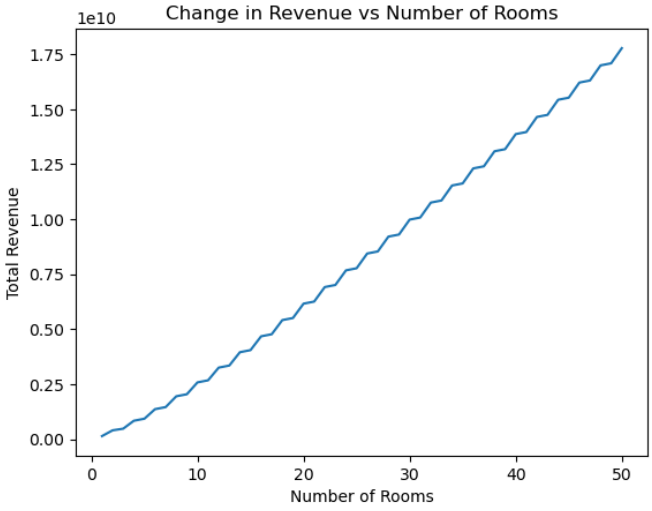
The objective value of the schedule generated for this specific case study after removing the diversity constraint was $209,334$ dollars in total.

5 Experiments

After doing some analysis on our case study, we shall test how our model behaves with different parameters. Now, we will have 360 times periods, where each time period is 2 hours, similar to what we had before. However, we know extend the time to a month. Furthermore, we now have 250 arbitrary movies. The rental cost is uniformly distributed between 10,000 and 100,000, and the starting time is also randomly distributed between the entire month. The predicted revenue will start between 2000 and 500,000, and decays as the same function as in the case study. The distribution of the data is shown in the Appendix.

5.1 Number of Screens

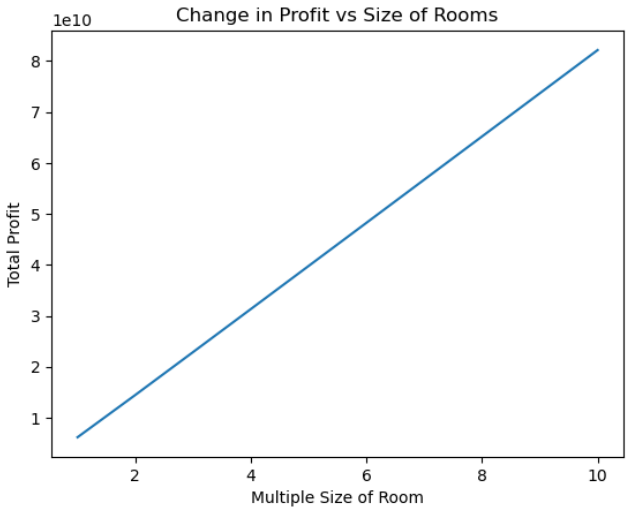
We shall first determine whether there is a direct correlation between the number of screens as well as the profit a movie theatre makes.



We first realize that there is a linear relationship between the number of rooms and the total revenue, which makes sense, since this leaves opportunities for more movies to play. What's interesting is that the line isn't strictly linear, and there are bumps which indicate that at some points, adding 1 movie theatre is not as optimal.

5.2 Sizes of Movie Rooms

We then test the impact of the sizes of the room on the total revenue. Since our model states that the revenue generated by a movie grows at a logarithmic scale, we shall test the impact through different multiples.

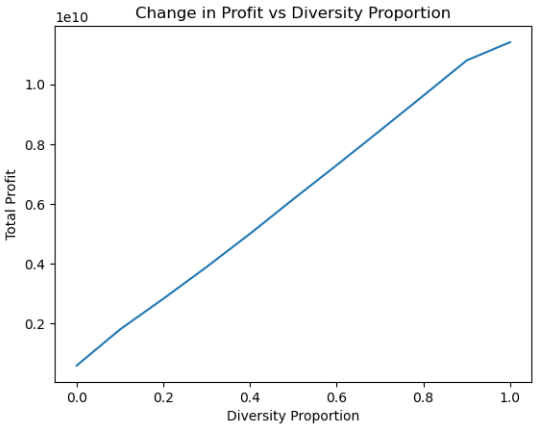


As expected, we observe a perfectly linear relationship between multiplying the size of a room and the total profit.

Furthermore, we then experiment with half of the rooms doubling in size versus all of the rooms increasing in 50% the capacity. Increasing the sizes of all the rooms led to a profit of 10,275,598,545 whereas increasing the sizes of half the rooms led to a profit of 13,613,686,490, which is 32.49% more. Hence, this is the reason you are more likely to see rooms of different sizes of a movie theatre.

5.3 Effect of Diversity

In our formulation, we added a constraint, where movie theatres determine how many screens a movie can take up in a movie theatre, since the optimal solution is more often to stream the most popular movie on all the screens. Our analysis here determines how much the proportion affects the final revenue.



Similarly to before, there seems to be a linear relationship between the diversity proportion and the revenue, which makes sense, since we are working with linear programs.

5.4 Buying vs Renting

In our formulation, we assume that all the movie theatres rent movies, and then it costs money for each time period to rent a movie. In this section, we compare the difference in which movie theatres buy movies rather than rent movies. It will cost more to buy movies, but if they do rent it, they get it keep it for the entire time period.

5.4.1 Reformulation of the Problem

Here, the total revenue does not change since we still decide which movie plays on what screen at a given time. However, the costs now change since it is a one-time cost. This gives us our new objective function. Let z_i be a binary variable that is 1 if movie i is ever played and 0 otherwise. Furthermore, let BC denote the cost of buying the movie. This gives us our new formulation:

$$\max \quad \sum_{i,j,k} x_{i,j,k} r_{i,j,k} - \left(\sum_{i \in [M]} \text{BC}_i \cdot z_i + N \cdot \text{OC} \right)$$

$$\begin{aligned}
\sum_{i \in [M]} x_{i,j,k-b_i} &\leq 1, && \forall j \in [S], t \in [N] \\
\sum_{i \in [M]} \sum_{j \in [S]} x_{i,j,k} &\leq S \cdot N \cdot z_i && \forall k \in [N] \\
\sum_{j \in [S]} x_{i,j,k} &\leq \lfloor p \cdot S \rfloor && \forall i \in [M], k \in [N] \\
x_{i,j,k} &\in \{0,1\} \\
y_{i,k} &\in \{0,1\}
\end{aligned}$$

5.4.2 Results

We summarize the results of our new simulation in the table below:

	Profit	Number of Unique Movies
Renting	\$5,726,053,327	228
Buying	\$4,767,736,137	114

As observed, movie theatres tend to make more profit by renting movies rather than simply buying. Furthermore, the algorithm was able to schedule a higher diversity of movies by renting them, which makes sense, since by buying them, we tend to use the movie that is the most popular again and again.

Hence, through our analysis above, we claim that it is much more profitable for movie theatres to rent movies rather than buy them.

6 Discussion

6.1 Limitations

While our model does create outstanding results and propose answers to several different questions, there are several limitations that we need to consider. For one, our input parameters are mostly estimated, since such data is not available to the public. We also don't take into account other influencing factors, such as the time of day, the seasonality, or certain holidays. Furthermore, our sources of revenue and cost are highly limited. Our costs do not consider any retail/merchandise sales or individual rental services. Our revenue also doesn't take into account the revenue is typically split with studios that actually produce the movies.

Most of these limitations stem from the fact that much of the data is unavailable online. However, the basis of our model still stands. Given enough data, the model is still usable, and only certain changes need to be made in order to account for the new parameters.

6.2 Conclusion

In this project, we formulate the movie scheduling program as an integer program. We consider different factors such as a movie's popularity as well as different costs which outputs the optimal scheduling of movies in different movie screens to produce the most revenue. We introduce an idea of using machine learning techniques in order to predict the popularity of a movie which allows us to simulate our model using code. We then perform a case study on 3 movies on a movie theatre to analyze our results, which we then apply to a much larger simulated dataset where we can answer several questions, including the relationship between the number of screens and the revenue as well as why movie theatres tend to rent movies rather than buy them.

Some future approaches might include fine-tuning the popularity prediction model, obtaining more data for more accurate parameters, as well as consider new models such that movie theatres can decide when to start playing movies as well as consider the different seasonalities.

References

[1] Edet, S. Optimal movie schedule and prediction.. *By Samuel Edet :: SSRN*. (2017,7), https://dx.doi.org/10.2139/ssrn.2996512

[2] Eliashberg, J. Demand-driven scheduling of movies in a multiplex. *International Journal Of Research In Marketing*. (2009,4), https://www.sciencedirect.com/science/article/pii/S0167811609000160

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L. & Polosukhin, I. Attention is all you need. *ArXiv.org*. (2023,8), https://arxiv.org/abs/1706.03762

Appendix (Code)

```
'''
M: Number of Movies
S: Number of Screens
N: NUmber of Time Periods
'''

from gurobipy import *

M = 100
S = 20
N = 360

# Simulate how much revenue movie i make on screen j on the kth time period

r = {}
for i in range(M):
    r[i] = {}
    for j in range(S):
        r[i][j] = {}
        for k in range(N):
            r[i][j][k] = 0  # Initialize with 0

for i in range(M):
    init_val = temp[i]
    for j in range(S):
        for k in range(N):
                r[i][j][k] = int(init_val/(k/10+1))

model = Model()
model.Params.LogToConsole = 0

# Decision variables
x = {}
for i in range(M):
    x[i] = {}
    for j in range(S):
        x[i][j] = {}
        for k in range(-N, N):
            x[i][j][k] = model.addVar(vtype=GRB.BINARY, name=f'x_{i}_{j}_{k}')

y = {}
for i in range(M):
    y[i] = {}
    for k in range(N):
        y[i][k] = model.addVar(vtype=GRB.BINARY, name=f'y_{i}_{k}')


# Objective function
obj_expr = quicksum(x[i][j][k] * r[i][j][k] for i in range(M) for j in range(S) for k in range(N))
obj_expr -= quicksum(RC[i] * y[i][k] for i in range(M) for k in range(N))
```

```
obj_expr -= OC * N


model.setObjective(obj_expr, GRB.MAXIMIZE)

# Constraints
for j in range(S):
    for k in range(N):
        model.addConstr(quicksum(x[i][j][k-b[i]] for i in range(M)) <= 1, f'constraint1_{j}_{k}')

for i in range(M):
    for k in range(N):
        model.addConstr(quicksum(x[i][j][k] for j in range(S)) <= S * y[i][k], f'constraint2_{i}_{k}')


for i in range(M):
    for k in range(N):
        expr = quicksum(x[i][j][k] for j in range(S))
        model.addConstr(expr <= S/2+1, f'new_constraint_{j}_{k}')


model.optimize()

# Print the results
if model.status == GRB.OPTIMAL:
    res = []
    s1 = s2 = s3 = s4 = s5 = s6 = []
    print("Optimal solution found!")
    print("Optimal objective value:", model.objVal)
```