Dynamic Programming for Movie Scheduling

Pawas Gupta PAWAS.GUPTA@OPERASOLUTIONS.COM
Advanced Solutions, Express Trade Towers, Film City, Noida 201301
Praveen Kumar Omar PRAVEEN.OMAR@OPERASOLUTIONS.COM
Advanced Solutions, Express Trade Towers, Film City, Noida 201301

Abstract

The aim is to find movie schedule from given set (of all possible schedulable shows), which maximizes the profit while satisfying some set of given constraints. In this paper, we propose an optimization algorithm based on Dynamic Programming to achieve the optimization objective. The task of achieving optimal solution is challenging because of certain factors: 1) Large input search space for the optimizer to solve. 2.) There are many linear constraints which need to be handled during optimization process. 3.) There are certain nonlinear coefficients that are introduced in system while solving for optimality, which increases time and space complexity. And most crucial aspect is, all of this needs to be achieved in limited time frame and with limited memory available.

1. Introduction

Opera is developing a solution for a cinema chain based in United Kingdom. VUE is 3rd largest cinema chain in United Kingdom, with cinema multiplexes at more than 84 locations, capturing around 22% of the market. VUE's multiplexes vary in size from 4 to 30 screens, with a total of around 800 screens across the country. All the multiplexes are managed by planning group committee, whose members manually decide the movie schedule for a given week, which will be on sale for a given multiplex.

The mentioned product is an automated scheduling system, which predicts the behavior of movies playing in an upcoming week, and then prepares a schedule for playing those movies in a way, that maximizes the revenues earned from the ticket and retail sales of movies played. The system has various phases, which it goes through before churning out final solution. They are: Constraints Extraction, Prediction, Sample Space Generation and finally; Optimization with Constraints satisfaction. One of the elaborate papers on movie forecasting and scheduling is by (Jehoshua Eliashberg, 2009) After predicting behavior of a movie for a day and for time of day, a large sample space is generated which is set of all possible combinations of a given movie that can be played at a location on a given screen at given time of a given day.

The job of optimization process to find the optimal selection from given sample space, which maximizes the profit while adhering to given linear and nonlinear time constraint set. In order to generate a movie schedule, optimizer has to consider impact of several factors at play; age and duration of movie, prints available for a movie, format of movie (2D, 3D) etc. It also has to factor in the operational costs of running and maintaining a cinema multiplex. Apart from all this, the algorithm should be such that it can be easily scaled across multiple sites, and should be capable of being generalized across different cinema houses. And finally, if not the least, the implemented algorithm should not be computationally expensive, generating optimal solution in short span of time. In this paper, we would like to discuss approach used in implementation of optimization process in our system; Dynamic Programming to find most profitable schedule, while taking into consideration all above listed factors.

1.1 Introduction to Dynamic Programming Algorithm

Dynamic programming is a method for solving a complex problem by breaking it down into a collection of simpler sub problems, solving each of those sub problems just once, and storing their solutions - ideally, using a memory-based data structure. The next time the same sub problem occurs, instead of re computing its solution, one simply looks up the previously computed solution, thereby saving computation time at the expense of modest expenditure in storage space. The act of storing solutions to sub problems is called "**memoization**". (Farmer, 2008)

There are two major characteristics of a DP problem: the optimal solution can be constructed from optimal solution of its sub problems (optimal structure); and there exists some places where we solve same sub problem more than

The Fourth Internal Technical and Analytical Conference of Opera Solutions (TACOS 2016), March 3rd, 2016, San Diego, USA.

once. A dynamic programming algorithm will examine the previously solved sub problems and will combine their solutions to give the best solution for the given problem

There are two ways of doing this:

- Top-Down Approach: Start solving the given problem by breaking it down. If you see that the problem has been solved already, and then just returns the saved answer. If it has not been solved, solve it and save the answer. This is usually easy to think of and very intuitive. This is referred to as **Memoization**.
- Bottom-Up Approach: Analyze the problem and see the order, in which the sub-problems are solved and start solving from the trivial sub problem, up towards the given problem. In this process, it is guaranteed that the sub problems are solved before solving the problem. This is referred to as Dynamic Programming

2. DP Application for movie scheduling

The optimization objective is achieved with the help of Dynamic Programming technique. The movie schedule for any given week is created at Site X Day level, with schedule for each day for a site being independent of schedule for any other day, and a set of schedule for all seven days for a site, is the weekly schedule for that site. Input search space for any Site and Day combination, consists of all the combinations at Screen X Movie X Session Start Time level, and the algorithm finds the best solution from it. While finding the best movie to play at a certain time on a certain screen, it has to consider number of prints available for that movie, number of minimum or maximum shows of that movie to play, any other show of the same movie or any other movie playing in any other screen at similar time, when the screen opens or closes etc. All these factors are constraints which impacts the optimization objective.

Current levels of constraints are at Site X Day level, which allows scheduling for any Site and Day to be independent from schedule generation of any other Day on Site.

2.1 Optimization Approach Sequence:

For generating complete schedule for a Site X Day, we achieve this by generating schedule for each screen at site (one at a time), for a given day. Schedule for each screen is generated using Dynamic Programming Based Optimization Algorithm, applied to each screen in a sequential fashion. Once a screen has been completely scheduled, the schedule against it remains fixed. Hence, the order in which screens are scheduled might impact quality of schedule. To mitigate the effect of this, certain screen heuristics are followed, which ranks the screens in certain order prior to scheduling

We generate two set of schedules with two different screen ordering, one being from largest size screen to smallest size screen and other being based on a heuristic assigning a bit lower ranking to 3D screens(in an attempt to boost scheduling of 3D format films). Once both set of schedules have been generated for each Site X Day for a given week, they are evaluated and the most profitable schedule for each day among both set of schedules becomes the part of final schedule for desired Site X Week.

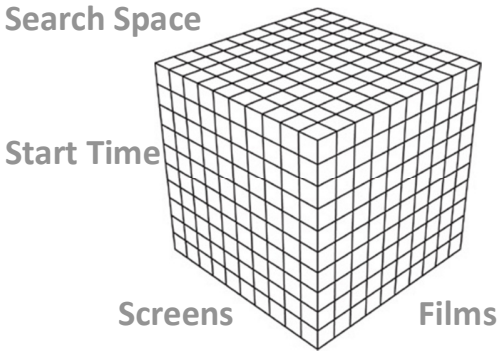

Fig 2.1.1: Search space repersentation

2.2 Optimization Problem

Suppose there are **N** numbers of schedulable time slots and for each slot, there are **F** numbers of schedulable films. Hence, the objective would be to find the most profitable set of sessions out of **N** X **F** schedulable sessions for given screen, where:

N: total number of 5 minute time slots where a movie can start

F: total number of movies which can potentially start at a given 5 minute time slot in accordance to rank).

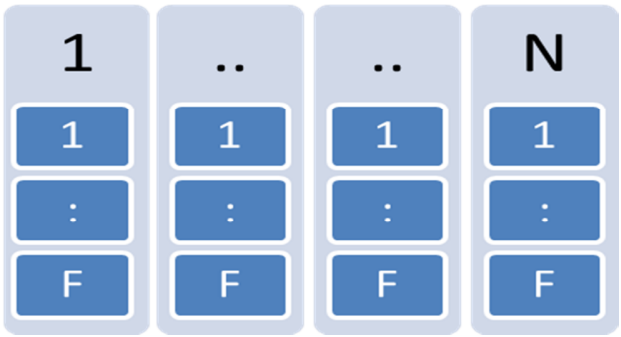

Fig 2.2.1: N X F matrix for all slots and screen combinations

2.3 Solution Framework

- **Base Case**

  If **N** = 1, most profitable schedule will be created by scheduling a film (out of **F** number of films) giving maximum profit among all films. Hence, if a screen's schedule is starting at last slot i.e. slot = **N,** then

  $$schedule[slot = \boldsymbol{N}] = MaxProfitSchedule(S(N,1), S(N,2), \ldots., S(N,F))$$

  Where **S(N,1), S(N,2)**, …. , **S(N,F)** are schedules created by scheduling of different films at last slot (slot=**N**).

- **General Case**

  If a screen's scheduling is allowed to start on or after slot ***t***, then:

  schedule[slot=t]=MaxProfitSchedule{MaxProfitSchedule(S(t,1),S(t,2), …. , S(t,F)), schedule[slot=t+1] }

  Where **S(t,1), S(t,2), …. , S(t,F)** are schedules created by scheduling of different films at slot=**t**.

  And for any movie ***i***, **S(t,i)** is created as below:

  $$S(t,i) = \begin{cases} [Movie\ \boldsymbol{i}\ at\ Slot\ \boldsymbol{t}] + schedule[slot = (t + lengthMovie(i))], if\ (t + lengthMovie(i)) \le N \\ [Movie\ i\ at\ Slot\ t] \quad , Otherwise \end{cases}$$

  Essentially, schedule[slot=***t***] is providing the most profitable schedule starting at slot=***t***. In order to do that, it tries to schedule each film at slot ***t*** followed by already best schedule at ***t*** + lengthMovie() for respective movies. Then, it chooses the schedule with maximum profit (if a movie were to start at slot ***t*** itself) and compares it with the profit of best schedule at slot ***t+1*** i.e. schedule[slot=t+1], and assigns the more profitable schedule to schedule[slot=t]. DP adds value here by storing best schedule for all slots > ***t*** and by using them for generating **S(t,i)** as required.

- **Result Case**

  If a screen's scheduling is allowed to start on or after slot *1(desired case)*, then

  $$sschedule[slot = 1] = MaxProfitSchedule\{ MaxProfitSchedule\ (S(1,1), S(1,2), \ldots., S(1,F)), schedule[slot = 2] \}$$

  Essentially, schedule[slot=***1***] is the optimal schedule for screen and exit point of DP based algorithm for schedule generation of a screen.

  Profit Computation for ***S(t,i)***:

  $$Profit(S(t,i)) = MarginalProfit(Movie\ i\ at\ slot\ t) + Profit(schedule[t + lengthMovie(i)])$$

3. Constraints satisfaction while DP scheduling

While creating schedule for a screen, a Film **i** is considered for scheduling at slot **t** only if it doesn't violate any of the following constraints with the already schedules screens.

3.1 Prints Constraints

If an extra print used for scheduling of Film ***i*** at slot ***t*** leads to violation of Prints used across already scheduled screens, ***S(t,i)*** is dropped from consideration for finding best schedule at slot=t

3.2 Spacing Rule

If the Film being scheduled violates spacing rule i.e. if there are any sessions within prohibited range of spacing rule (applicable to Film ***i*** based on its available prints), ***S(t,i)*** is dropped from consideration for finding best schedule at slot=t

3.3 Special Shows

In order to force scheduling of special sessions in DP phase automatically, sessions having their start time or end time within run duration of special shows are dropped from scheduling i.e. sessions of type 1 to 4 are dropped as illustrated below.

3.4 Min show Constraints

Min Show constraints are defined as such where certain minimum number of shows of a movie has to be played at a site and day, during a given time frame. These constraints are a very important part of movie scheduling

system and how the final schedule looks and performs, depends a lot these constraints.

3.4.1 Min Show Constraints Satisfaction Overview

The current optimization system is not designed in a manner to handle these constraints via dynamic programming, so these constraints are handled outside schedule optimizer block (implemented using dynamic programming.)

In this phase, an already optimized schedule is passed with a set of Min Shows constraints to be satisfied. Since we are trying to satisfy constraints after optimization step has been completed, there will be some loss incurred due to forced satisfaction of the constraint, as it will lead to a suboptimal solution. So, the current min show satisfaction algorithm is implemented such that it minimizes the loss in profit while trying to satisfy maximum possible min shows constraints

Current algorithm checks for sessions (in already optimized schedule) which already satisfy some of the min shows, tags them as min shows satisfying sessions and update remaining min show constraint to be satisfied.

Algorithm For unsatisfied min show scheduling

Input: Unsatisfied min show constraints C, Number of unsatisfied min show constraint U

Repeat

While (U > 0)

Find a Screen X Slot combination within constrained interval of C, such that scheduling a Show for C doesn't:

- Violate Prints Constraints or Spacing Rule with rest of the schedule
- Doesn't try to remove any special sessions or some min show satisfying session
- Causes minimum marginal loss in the profit of the schedule before insertion and deletion of appropriate sessions for satisfying constraint C

If (True)

- Delete necessary sessions from the schedule and insert a show satisfying min shows constraint C.
- Tag that shows as a min show satisfying sessions so that it isn't attempted for removal in rest of the process.
- U= U-1

Else

Break

Essentially, min shows satisfaction is attempted in a sequential fashion and once a session start playing role in satisfaction of any min show constraint, it remains fixed. Though this strategy tries to ensure minimum loss in profits but may not guarantee satisfaction of all min show constraints. In order to minimize runtime complexities, constraint satisfaction is attempted for a certain sequence (based on a heuristic in an attempt to maximize number of constraints satisfied) of constraints only.

3.4.2 Screen specific Min Show satisfaction using Vibrating Min Show Approach

The current schedule generation process does not ensure the satisfaction for screen specific min show constraints. To ensure their satisfaction, different min show satisfaction approach is used, where no show is fixed on the screen until all min show constraints have been satisfied on that screen (If feasible).

In this, we will continue min show satisfaction as per above discussed algorithm until we find a constraint which is not schedulable at any slot in given time range. Then, we will backtrack one step and re-schedule the immediate previous show to their next best slot (in terms of loss to the schedule). This will be iterated to next best slots unless constraint is satisfied or the immediate previous show runs out of slots. In later case, the previous of the immediate previous show will need to move to its next best slot. This process goes on recursively until we find slots for all the shows. It will fail if these constraints are not feasible at all on this screen.

3.4.3 Re-Run DP Phase

Min Show constraints satisfaction leads to gap in schedule schedule due to deletion and insertion at various places. Hence, in an attempt to address such gaps and improve the profit number from schedule:

- Pin or fix sessions which are Special Shows or play a part in satisfaction of any min shows constraints
- Using above pinned sessions as base, create remaining schedule for the Site X Day using Dynamic Programming based scheduler again

So wherever possible, DP will try to improve the profit of the schedule assuming pinned sessions are already present. This phase provides most profitable schedule for Site X Day which satisfies significant number of min shows constraints.

3.5 Site Flow Constraints

At any site, where people go and watch a movie, the people assemble in auditorium. The auditorium fills up whenever a movie is about to start or whenever a movie is about to end. But, the space of the auditorium is finite and can accommodate only a finite number of people at any given time. So, for a site and day combination, client gives us a constraint such that for any scheduled show starting at time **t** across all the screens, sum of combined attendance of all the shows starting between **t** and **t + I,** and combined attendance of all the shows ending between **t** and **t + I** cannot exceed a certain defined number, which is site flow constraint, where **I** is interval duration

This constraint cannot be satisfied while generating schedule as DP runs screen by screen and we cannot determine which movies are going to be scheduled in subsequent screens hence, will affect the shows on screen being currently scheduled. This leaves us option to look for site flow violation once the schedule has been generated. For this we start finding slots on which site flow is violated using the formula described above and if on a slot site flow is violated we remove all those shows form search space which are responsible for this violation and generate schedule again.

4. Why Dynamic Programming approach

Before optimization via dynamic programming was decided as final algorithm to be implemented for schedule optimization, many other algorithms were considered, with their merits and demerits and finally, this algorithm as final choice. Following are the approaches that were implemented and tested before dynamic programming solution:

- **Greedy algorithm**: It selects the best film x time x screen in a greedy fashion. Then, schedules the film at that time and screen. Blocks the appropriate search space to avoid screen overlaps and repeats until no more session can be added.
- **Gurobi algorithm:** Gurobi is an industrial strength 3rd party optimizer**.** It formulates the problem as mixed integer linear program (MILP). It Encodes constraints as well as objective function and converts nonlinear problem into piece wise linear function
- **Greedy + Gurobi Algorithm**: This algorithm uses solution of greedy optimization as starting point for Gurobi optimization. The idea is to restrict or reduce the input search space for Gurobi optimization, allowing the computation time to improve significantly.

4.1 Optimization Performance Matrix

	Greedy	Gurobi	Greedy + Gurobi	Dynamic Programming
Run Time	Around 1 hour	Takes 1 week or more	Around 3 hours	10 minutes
Constraint Satisfaction	Many constraints violation occurs, especially across min show constraints	Leads to satisfaction of all constraints	Leads to constraints satisfaction, 100% satisfaction of all min show constraints	Around 95% min show constraints get satisfied.
Convergence	Convergence is reached in given time frame	May never converge, also may lead to infeasable solution	May or may not converge depending on size of input	For a day, convergeas at site and screen level
Cost of running	Developed inhouse, no running cost	Gurobi is a third party industrail strength optimizer, having high run costs per hour (around $25)	Gurobi is a third party industrail strength optimizer, having high run costs per hour (around $25)	Developed inhouse, no running cost
Pros	Fast, predictable and Robust	Tries to converge to optimal solution, is able to handle any constraint	Better run time than Gurobi, converges lot faster than only gurobi optimization	Best run time of all optimization methods, schedule more profitable than combined Greedy and gurobi schedule
Cons	Cannot satisfy all constraints, more min shows detoriate schedule performance, and leaves gap in final schedule (fragmantation)]	Largest convergence time, may result in infrasable solution and may also lead to infeasablity if not converged.	Input depends on greedy solution. May or may not find best solution. Also, it may return infeasable solution if greedy schedule is poor	Cannot satisfy all constraints, and addition of constraints degrades profit.

The client expects the result to be available in a limited time frame and they expect system to handle multiple runs during a given day. Also, they want constraints to be satisfied as constraints are important for their business

processes and are included as part of terms and conditions negotiated with distributor (person who provide movie prints). Thus, Greedy or only Gurobi optimization algorithm are not used as they either leads to poor constraints satisfaction or take a lot of time to converge (well ahead of expected turnaround time). Also, running greedy or only Gurobi optimization for limited set of time, provides poor schedule with lot of fragmentation

Even though all the min show constraints are not satisfied by the dynamic programming based optimizer (around 5%), it leads to better profit results as compared to Greedy + Gurobi based optimization. Poor greedy optimized schedule results in poor Gurobi optimized schedules and the runtime of combined Greedy and Gurobi Approach is still around 3 hours, whereas Dynamic programming based algorithm generates solution within 10 minutes, allowing multiple runs to be scheduled within a day, which is seemingly impossible in any other optimization scenario.

5. Results

Fig 5.1: Snippet of schedule for a site and day combination, generated from DP optimization.

6. Summary

The current Dynamic programming based optimization algorithm is used to create most profitable schedule for a site and day by scheduling one screen at a time, and combined schedule for all the screens is optimal solution. In current approach, print constraints, special shows and spacing rule constraints are dynamically handled, while other constraints are not handled by DP block. Other constraints such as min show constraints, site flow constraints are not handled while optimizing for maximum profit and inclusion of any of these introduces profit losses. Thus their algorithms are such that they try to satisfy these constraints while minimizing losses incurred from them. Out of many approaches implemented in past, dynamic programming based optimization is the best fit with the current client requirements in which client expects system to generate results fast along with capability to initiate multiple runs within a day, even at cost of reduced constraints satisfaction.

7. Future Work

Modification in number and types of constraints

Currently we have min/max constraint having fix values of start and end range. A more evolved version of such constraint would be fuzzy start range and end range with 30 minutes window.

Introducing projector constraints

On a particular site there are several screens and screens have local hard drives to save the movies. Projectors play movies from these drives that have limited space. Projector Hard Drives are usually unable to hold all films required for a full week. Multiple transfers take place between the Projector and the site's central library server. There are limits to the transfer bandwidth into a Projector and out of a library server. Constraints should be included to ensure that schedules do not require file transfers that exceed X% of the maximum possible given the technical infrastructure at UK and Ireland Sites (the “Projector Constraints”). Users will be able to adjust X.

Acknowledgement

We would like to thank Alain Biem, Ritesh Aggarwal, Anurag Gupta and the entire reviewing committee for their guidance in making significant improvements to the paper. We would also like to thank Rahul Garg, Shubh Bansal, Utsav and Abhinn Kothari for their support in building the DP based optimizer.

References

Farmer, J. (2008). Introduction to Dynamic Programming. *20bits*.

Jehoshua Eliashberg, Q. H. (2009). Demand-driven scheduling of movies in a multiplex. *International Journal of Research in Marketing*.