Adaptive Nested Algorithms for Balanced Scheduling

Stetson Bost

Introduction

Scheduling is very important for maintaining order, for both individuals and larger organizations. In a fast-paced environment like Harvey Mudd, schedules can help many people stay organized and manage stress. With this project, we wanted to develop algorithms capable of creating effective and adaptable schedules.

Algorithm for Personal Scheduling

Our algorithm for creating a personal schedule aims to both accomplish all necessary tasks and maintain a balanced lifestyle. We treat a schedule as a collection of relatively short *time slots*, and contiguous time slots can form *blocks*. A group of consecutive blocks can form a *section*, identified by it starting block and ending block. The algorithm has three nested components.

1. Assign tasks to sections.
2. Assign tasks to blocks within sections.
3. Assign tasks to time slots within blocks.

Each of these steps takes into account various aspects of the tasks, including deadlines or available time windows, amount of time needed, and intensity. Additionally, each block has a minimum permissible amount of time reserved for non-work activities, which may or may not be structured, depending on user preference. With this non-work time and by maintaining as equal intensity levels as possible among blocks, this algorithm prioritizes a balanced schedule.

Task	Due Date	Duration	Intensity
Task 1	Monday	6 hours	0.3
Task 2	Friday	3 hours	0.6
Task 3	Thursday	4 hours	0.5
…	…	…	…

(a) Start with a list of tasks that need to be completed.

Section: Tuesday 8 AM - Midnight

Block 1	8 AM - 12 noon
Block 2	12 noon - 6 PM
Block 3	6 PM - 11 PM

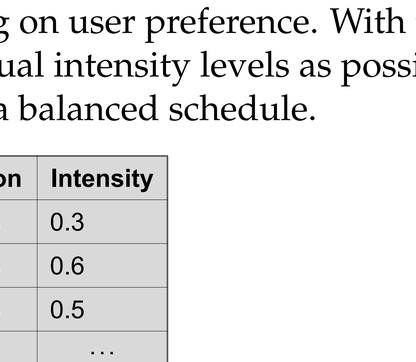
Non-Work Time	4 hours
Task V	2 hours
Task W	1 hour
Task X	2 hours
Task Y	5 hour
Task Z	2 hour

(b) Create a section made up of time blocks, and assign tasks to the section. Non-work time is treated as a task.

Block 2: Tuesday 12 noon - 6 PM

Non-Work Time	2 hours
Task X	2 hours
Task Y	1 hour
Task Z	1 hour

(c) Assign tasks to each block in the section.



(d) Schedule tasks within the block.

Figure 1: Example for personal scheduling algorithm. Given a task list, assign tasks to sections, then to smaller blocks, then to time slots on the schedule.

Algorithm for Event Scheduling

Members of a group often have to schedule events where a number people must attend. We wanted to create an algorithm capable of scheduling events with multiple attendees.

The event can be either recurring ("regular") or non-recurring ("special"), has restrictions on available times, some desired or required attendees.

Probabilistic Approach

Using historic data, we can use a probabilistic version of the k-nearest neighbors algorithm to determine the likelihoods that all potential attendees will attend a specific event if it is scheduled at some potential times (Holmes). By summing these likelihoods across all attendees, we are able to get the expected value for the number of people at the event.

TODO Title (Approach 2)

This approach make use of the Personal Scheduling algorithm detailed to the left with some slight modifications.

Future Work

To further validate this work, it will be useful to test the algorithms on actual scheduling problems. For the personal scheduling algorithm, it will be helpful to collect schedule data from individuals who are interested in applying the algorithm to their own schedules. Their feedback can be used to evaluate and revise the algorithm to better suit their needs preferences. For event scheduling, data would need to be collected from an organization, preferably one that is willing to use the algorithm to schedule their events. Additionally, for an organization that uses a common calendar application, it would be interesting to integrate the event scheduling algorithms with the calendar to automatically schedule events.

References

C. C. Holmes and N. M. Adams. *A probabilistic nearest neighbour method for statistical pattern recognition* (2002). `http://hedibert.org/wp-content/uploads/2016/02/holmes-adams-2002.pdf`

For Further Information

For more information about this project, please contact `sbost@hmc.edu` or `gu@hmc.edu`.

Acknowledgments

I would like to thank Professor Weiqing Gu for her guidance throughout this research. I am also very grateful to the Rose Hills Foundation Science & Engineering Summer Undergraduate Research Fellowship program for their kind and generous support of this project.

Advisor: Weiqing Gu