Adaptive Nested Algorithms for Balanced Scheduling

Stetson Bost
Department of Mathematics
Harvey Mudd College
Claremont, CA 91711
sbost@hmc.edu

Abstract

Scheduling, adaptive algorithm, nested algorithm, machine learning, big data

1 Scheduling Problem

Scheduling is a very large component of modern life. In a fast-paced world, it can be important to effectively plan one's schedule in a way that is conducive to productivity while prioritizing the importance of a healthy balance between different types of activities.

We aimed to develop algorithm that create balanced schedules that can be adapted to person to prioritize his or her needs.

2 Background and Motivation

The primary motivation for pursuing this work was the perpetually active and hectic environment of a college campus, primarily with Harvey Mudd College in mind. Students are faced with many academic obligations, from doing homework to attending class to studying for exams. While focusing on academics is important and necessary, it is also valuable to have a healthy balance of different activities to support a student's personal wellbeing. There are many ways for students to arrange their schedules, but rarely does a "one size fits all" approach work for creating students' schedules. Each person has a unique style of working and individual needs, so it makes sense that a person's schedule should be tailored to fit the person.

Additionally, schedules are inherently dynamic. Events can be scheduled and canceled. One may occasionally have to deal with an emergency that preclude all other scheduled activities. Priorities can change.

For the remainder of this paper, we will use the terms *events* and *tasks* interchangeably to refer to an activity of some duration that should be added to a schedule.

3 General Approach

To address scheduling in a way that took into account balance and personal preferences, we tried to mimic human intuition when possible. Humans seem to intuitively prioritize tasks, but the ways that they do so can vary between different contexts and from person to person. Therefore, we wanted to create algorithms that were *adaptive*, in two senses of the word. First, we wanted something that could take in a stream of tasks to schedule and *adapt* according to when tasks are introduced. Second, we needed an algorithm that could *adapt* to an individual, taking into account the person's preferences and way of working.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

We decided to use *nested* algorithms that assigned events to a schedule in stages, where earlier stages schedule events more generally (i.e. the week it will be scheduled), while the later stages arranged events more specifically (i.e. the exact time when the event should take place).

4 Algorithm for Personal Scheduling

5 Algorithms for Event Scheduling

5.1 Probabilistic Approach

Suppose we know the probabilities for all potential attendees that they will attend all possible events. This means we can know how likely they are to attend a new event if it is scheduled for a particular time slot. Let A be the list of attendees, t be a possible start time, $p_a(t)$ be the probability that an attendee a will attend the event if scheduled at time t, and $\mathbb{E}[t]$ be the expected value of attendees at time t. Notice that for any particular start time t, if we sum all potential attendees' probability of attending, we will get the expected value of attendees at t. For each potential attendee a, go through all possible start times t, add the probability $p_a(t)$ of a attending at t to the value in the index corresponding to t. That is,

$$\mathbb{E}[t] = \sum_{a \in A} p_a(t).$$

If there are m possible start times, then we can create a $1 \times m$ matrix $\mathbb{E}$ of the expected number of attendees at each time.

$$\left[\sum_{a \in A} p_a(t_1) \quad \sum_{a \in A} p_a(t_2) \quad \cdots \quad \sum_{a \in A} p_a(t_n) \right]$$

6 Data

We have not yet used a real data set on which to test our algorithms. Historic data could be collected from the Harvey Mudd community through a combination of techniques such as online surveys and in-person interviews.

7 Examples of Scheduling Problems

There are many examples of personal scheduling problems that could be solved using similar algorithms.

One example, which will be explored later in this paper, is a college student making a weekly schedule that includes classes, homework, meals, free time, a job, sleep, etc.

Another example might be an (office) employee's work schedule. This can be considered on different time scales. For a day, the employee might need a schedule that includes when he or she should start and end, what time different tasks or meetings will take place, and when to have meals and breaks. With a longer timescale, such as a month or quarter, the employee might need to schedule which days to work on different projects, when training or business trips will take place, and when to take personal time off from work.

An elementary school teacher has required standards that his or her students must meet. The teacher must create a flexible schedule for the academic year that plans when to cover different topics in order to meet the requirements. Additionally, the teacher may need to plan for field trips, "slack" days (in case the class falls behind schedule), as well as incorporate a larger school-wide schedule including events such as assemblies, parent-teacher conferences, and breaks. The teacher may divide the topics into units, and the calendar into months or weeks. Depending on the pace of the class throughout the year, the schedule may need to be adjusted to either move faster or slower later on.

A family on vacation may have a list of activities they want to do while visiting a particular place before they leave.

The organizer of a conference may have a list of speakers that are going to give presentations. The organizer wants to schedule the speakers and breaks (for snacks, meals, etc.) such that the audience members generally remain engaged throughout the conference.

8 Goals for Schedules

Make sure everything that is necessary gets done by the time it must be completed.

Make sure there is some reasonable/healthy amount of “slack time” of personal time (if applicable).

Want to design a cost function that can measure how “balanced” a schedule is

When some cost function is applied to some time period (day, week, etc.), we want to minimize variance in the total cost among the time periods.

9 Algorithm

Each task has some time frame in which it must be completed and an associated intensity level corresponding roughly to the estimated amount of (mental) concentration needed to (effectively) perform the task.

A schedule is made up of timeslots. For our algorithm, a *timeslot* refers to the smallest period of time the algorithm will be dealing with. This can be 1 minute, 15 minutes, half an hour, 1 hour, etc. The schedule is preliminarily divided into blocks of time. (For our example, we use four- to five-hour blocks.)

There may be some tasks that have to occur at specific times (i.e. meetings, classes, etc.) regardless of other tasks in the schedule.

The algorithm consists of three main parts.

1. Distribute the tasks into sections of the schedule (on a scale larger than blocks), where all tasks within a section will be completed by the end of section.
2. Assign tasks in each section to a block.
3. Schedule tasks within each block.

9.1 Tasks to Sections

9.2 Tasks to Blocks

9.3 Tasks to Timeslots

10 Personal Example

Acknowledgments

References