CS358: Applied Algorithms

Final Project: Running Training Plan MIP (due 12/10/24)

Instructor: Sam McCauley

by: Rick Yanashita and Luke Zanuck

Introduction

Our project is a MIP model to optimize the training schedule for a runner preparing for a future race. The problem we are solving is how to efficiently allocate training workouts (with varying types, intensities, and constraints) in order to maximize the runner's fitness level at the end of the training period while accounting for factors such as rest days, consecutive workout days, strength training, and race requirements.

The runner has a set of available workouts, each with specific costs and fitness returns, as well as constraints on workout scheduling, rest days, and the need for strength training. The objective is to determine the optimal training plan that maximizes fitness at the end of the training period, subject to the specified constraints. These constraints include ensuring adequate recovery time, maintaining an appropriate balance of workout intensity, and meeting fitness goals for the races.

The motivation behind this project lies in the real-world challenge of creating an effective training schedule that takes into account the need for both physical progression (fitness gains) and recovery (to avoid overtraining). Although running coaches exist, training built by coaches are often exclusive to runners who are a part of a team. By using LP and MIP techniques, we aim to find the most efficient and feasible training plan for any runner.

Our deliverables for this project include the LP models that are generated by a Python script, and the Python script itself. The script allows the user to adjust various parameters in the LP, such as length of the training plan, the rest day requirements, baseline fitness, number of races, and the inclusion of strength training. In this report, we will present and explain all parts of the LP, an brief review of the code for the script, and an explanation of the optimal output for one Python-script-generated LP. Through this, we will showcase how our LP adapts to different constraints and objectives to produce an effective training plan for the runner.

The Linear Program

The objective of this LP, as stated above, is to determine the optimal training plan that maximizes fitness at the end of the training block. Formally, this is written as:

$$\text{objective: maximize } f_n$$

where n is the last day in the training plan and the length of the plan as well. The other variables of the plan are:

- f_i: integer. fitness for day i
- e_i: binary. option to do a easy workout on day i
- m_i: binary. option to do a medium workout on day i
- h_i: binary. option to do a hard workout on day i
- s_i: binary. option to do a strength workout on day i
- $sc1_i$: integer. tracks the number of strength training sessions from past 5 weeks/35 days
- $sc2_i$: binary. tracks if there have been 5 or more strength sessions scheduled (9 is the max for a 35 day period). If $sc2_i == 1$, reward the runner with 30 fitness points.
- c_i: integer. the number of consecutive workout days without a rest day at day i.
- r_k: integer. fitness needed for the kth mid-season race.

Furthermore, the constraints are designed to reflect the relationships between the variables, such that the LP accurately models the problem and finds the optimal solution. These constraints incorporate the investment costs and rewards associated with each workout type, such as:

- rest day: costs 0 fitness and returns 0 fitness after 1 day.
- easy workout: costs 1 and returns 2 after a 1 day investment.
- medium workout: costs 3 and returns 6 after a 2 day investment.
- hard workout: costs 5 and returns 12 after a 3 day investment.
- strength training: costs 2 and returns 30 after completing 5 strength sessions.

The constraints that use the variables and workout costs/rewards to structure the plan are:

- fitness per day: for each day $i \in n$, the total fitness at the end of that day must be ≥ 0. This means that the fitness from day $i-1$ and fitness rewarded from past workouts must be $\geq$ the amount of fitness the runner will expend through e_i, m_i, h_i, and s_i. The linear equation: $f_i = 2*e_{i-1}+6*m_{i-2}+12*h_{i-3}+f_{i-1}-30*stc_i+e_i+3*m_i+5*h_i+2*s_i$.
- medium and/or hard workouts can not be scheduled back-to-back: $m_i + m_{i-1} = 1$, $m_i + h_{i-1} = 1$, $h_i + h_{i-1} = 1$, $h_i + m_{i-1} = 1$.
- for each race, $f_i \geq r_k$ (the runner doesn't gain fitness from races but recovers the fitness equal to the "investment" after a couple of days).
- max consecutive workouts = INPUT (input value decided by user through the script). $e_i + m_i + h_i + e_{i-1} + m_{i-1} + h_{i-1} + e_{i-2} + m_{i-2} + h_{i-2} + e_{i-3} + m_{i-3} + h_{i-3} <=$ INPUT.

- there must be at least three days between strength workouts: $s_i+s_{i-1}+s_{i-2}+s_{i-3} <= 1$.
- total number of easy workouts $\geq 0.7*$ total number of workouts: $e_1+...+e_n-0.7*\text{total} \geq 0$
- total number of medium workouts $= 0.2*$ total number of workouts: $m_1+...+m_n-0.2*\text{total} = 0$
- total number of hard workouts $\leq 0.1*$ total number of workouts: $h_1+...+h_n-0.1*\text{total} \leq 0$

Along with the above constraints, there are a few special and specific constraints to count the number of completed strength workouts:

- count the number of strength sessions in the past five weeks: $sc1 = s_{i-34} + ... + s_{i-1}$.
- if the number of strength days is greater than 5, set $sc2_i = 1$. $sc2_i = 0.2 * sc1_i$.
- make sure that strength sessions are only counted towards the fitness reward once (no double counting strength sessions): $sc2_i + sc2_{i-1} + sc2_{i-2} + sc2_{i-3} + sc2_{i-4} = 1$.

The constraints to count the number of completed strength workouts is non-trivial and requires the constraints above. We will explain how each of them works. First, the constraint to count the number of strength sessions in the past five weeks checks every binary variable s_i from the last 34 days and adds up the 1 values. The 35 day/five weeks limit was chosen because it allows for the constraint on $sc2_i$ to work properly. Since there must be at least three days between strength workouts, the maximum number of strength workouts assignable in a 35 day period will be 9 (unrelated, but this also means that for the plan to take strength sessions into account the plan must be at least 35 days). To continue, since the max number of strength workouts assignable is 9, the constraint on $sc2_i$ will be 1 if there are $5-9$ workouts completed, and 0 if there are less than 5. The logic here is that if the runner doesn't complete 5 or more workouts, they will not recieve the 30 fitness reward. The fitness gains from doing $5-9$ workouts is the same, because the benefit of strength workouts can plateau. Similarly, the constraints make it impossible for more than 9 strength sessions to be completed in the 35 day period because doing excessive strength training can lead to injury, and thus is not part of this LP (which is trying to optimize fitness and not injure the runner).

Other parts to the LP include the bounds, binary variable declarations, and integer variable declarations. For the bounds, fitness must be non-negative: $f_i > 0$. For binary variables:

- all workout options are binary: e_i, m_i, h_i, s_i
- $sc2_i$ is binary

. Integer variables are: $sc1_i$, c_i, and r_k.

Python Script to Generate Optimal Training Plans

To generate all lines of the linear program in the CPLEX format, we created a Python script that generates the .lp file in accordance to specific values for the LP that can be set by the user. Values that can be set by the user are: plan length, rest days, baseline fitness, and number of races. Ideally, the user will input the values that make sense for their training. We have decided that "doubles" –doing two workouts in a day– will only be allowed if the baseline fitness is > 50. This makes sense because doubles are usually only performed by experienced runners who have higher levels of fitness. Keeping these values in mind, the script will generate the objective function, constraints, and bounds. This script is useful because real-life training plans tend to be for a few months, meaning that an LP to solve this problem would be thousands of lines in length.

The generated LP file contains all the lines of the LP needed to find the optimal solution. Below are some example lines from *plan.lp*, which is a 35-day plan:

```
maximize f35

subject to
f0 = 690
f1 - f0 + 1 easy1 + 3medium1 + 5hard1 + 2s1 = 0
f2 - f1 + 1 easy2 + 3medium2 + 5hard2 + 2s2 - 2 easy1 = 0
f3 - f2 + 1 easy3 + 3medium3 + 5hard3 + 2s3 - 2 easy2 - 6medium1 = 0
f4 - f3 + 1 easy4 + 3medium4 + 5hard4 + 2s4 - 2 easy3 - 6medium2 - 12hard1 = 0
f5 - f4 + 1 easy5 + 3medium5 + 5hard5 + 2s5 - 2 easy4 - 6medium3 - 12hard2 = 0
easy1 + medium1 + hard1 <= 1
medium1 + hard1 <= 1
easy1 + medium1 + hard1 + 3off_1 <= 3
off_1 + off_2 + off_3 + off_4 + off_5 + off_6 + off_7 = 1
easy35 + medium35 + hard35 + easy34 + medium34 + hard34 + easy33 + medium33
+ hard33 + easy32 + medium32 + hard32 + easy31 + medium31 + hard31 + easy30
+ medium30 + hard30 + easy29 + medium29 + hard29 <= 10
medium2 + hard1 <= 1
medium2 + medium1 <= 1
hard2 + medium1 <= 1
hard2 + hard1 <= 1
s1 + s2 + s3 + s4 <= 1
sc1_1 - s1 - s2 - s3 - s4 - s5 - s6 - s7 = 0
easy1 + easy2 + easy3 + easy4 + easy5 + easy6 + easy7 + easy8 + easy9 + easy10
+ easy11 + easy12 + easy13 + easy14 + easy15 + easy16 + easy17 + easy18
+ easy19 + easy20 + easy21 + easy22 + easy23 + easy24 + easy25 + easy26
+ easy27 + easy28 + easy29 + easy30 + easy31 + easy32 + easy33 + easy34
+ easy35 - 0.7total >= 0

bounds
f1 > 0
```

```
f2 > 0
f3 > 0
f4 > 0
```

The script will be included as part of the full submission, so we will not go in-depth about specific lines.

When actually running the script, generating a .lp file, and using glpk to solve the linear program, we experienced that plans longer than 35 days took very very long to run and find the optimal solution, so for any plans with length > 35, we split the .lp file into chunks and use the output of the previous chunk as input to the next. This optimization decreases the overall runtime of GLPK. We still ran a longer plan, with a plan length of 315, 1 rest days, baseline fitness of 50, and 13 races (this LP was split up). The output files of the long plan can be found in the repository as plan1.out - plan9.out.

Some Results from Running the Linear Program

We will briefly examine the results of plan.out, which was the LP for a 35 day training plan. The baseline fitness for this plan was 129, and the optimal solution found maximized the fitness such that the runner had 207 fitness by the last day. In this optimal solution, there are many "doubles", meaning the plan suggests the runner two do 2 workouts in a day. To balance out the use of fitness, the plan also recommends a lot of rest days for the runner. We found the use of frequent "doubles" interesting and validating, because "doubles" are use by runners to increase training load without adding excessive strain on their bodies. In real life, it is a healthy way to increase a lot of fitness, and it seems like it is the optimal strategy for this LP. Additionally, the plan suggests to do strength workouts on days $1, 9, 18, 22$ and 26. The strength sessions are spread out over 25 days, and the plan only performs 5 sessions (makes sense due to plan length).