Book Scanning - Specification

There are N libraries, each with their own signup time, set of books and number of books they can process per day. Every book is given a score, and having that book scanned in the end gives its respective score. Duplicate books give no extra points and different libraries may have the same book in their inventory. We can only have one library signing up at a given time, but libraries independently scan their books after the signup process.

Objective:

Given a description of libraries and books available, plan which books to scan from which library to maximize the total score of all scanned books, taking into account that each library needs to be signed up before it can ship books.

Sources

Official Problem Statement

Hashcode Competitors:

- HashCode 2020—how to reach a score in the top 5
- Google Hash Code 2020: How we took 98.5% of the Best Score

Problem Formulation

Solution representation: The libraries used in signup order and the respective books scanned from each one

Decision Variables: Score of the books to be checked in and the time it takes to sign in

Neighborhood / Mutation: Replace libraries chosen and/or the books scanned from each

Tabo List: Save the operations done (neighborhood) instead of the problem's state

Constraints: Only one signup at a time, operations after the deadline give no score

Evaluation: The score of all books in the library per day spent on signup

sum(book.score) / library.signup_time

Implemented Work ~ Input & Output

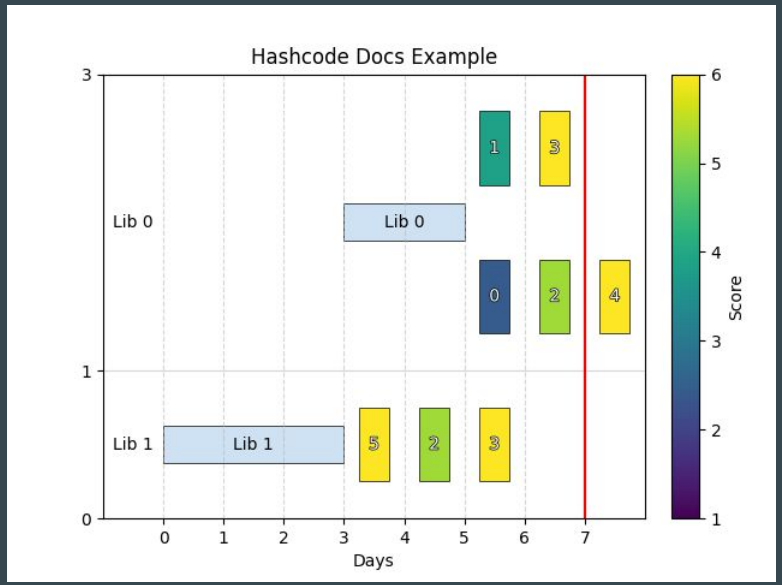
To obtain our solution we are using Python and pypy3. Python is used when we want to use our visualizer, otherwise pypy3 returns a solution in less than half the time. Our main data structure is a class (“Problem”).

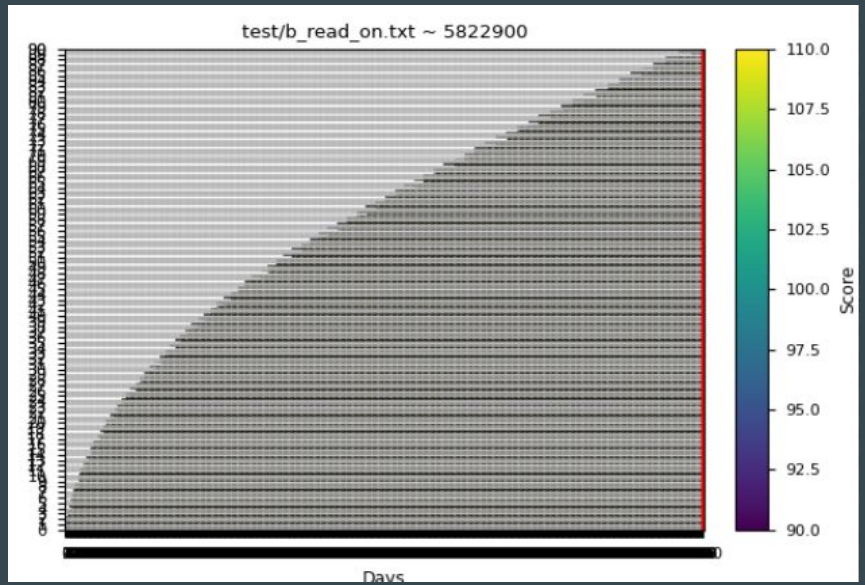
The first line of the input file contains three different integers that represent the number of different books, the number of libraries and the number of days in this order.

This is followed by L sections describing individual libraries from library 0 to library L–1. Each such section contains two lines: the first one contains the number of books in the library, followed by the number of days it takes to sign in the library ending with the number of books that can be shipped from the corresponding library daily. The second lines contains N integers that represent the ID of the library’s books.

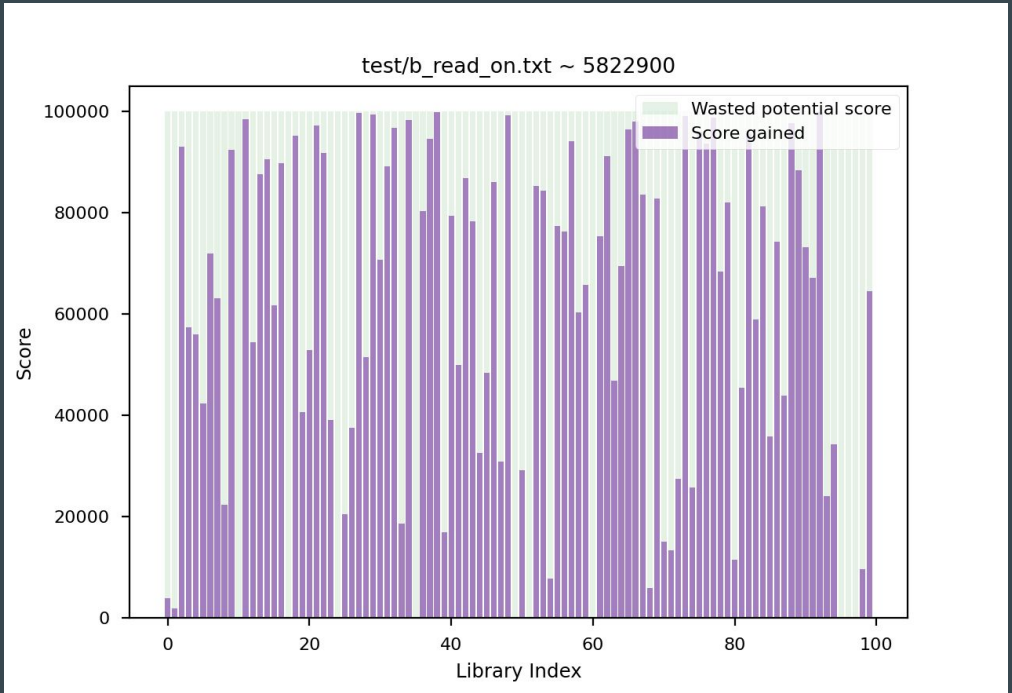
We also developed 2 different matplotlib visualizations to help with the solutions.

Problem Visualization ~ How NOT to do it (too slow)





Problem Visualization ~ Much faster method (still struggles with dataset D, which has 30k libraries)



Implemented Work

Currently we have implemented a parser for the input files and a way to input the solution to an output file. On top of that we have also implemented the algorithm Hill Climbing with steepest ascent. We only recalculate the value of each library every N percent days. This obtained some relatively good results (compared to other hashcode competitors) in a much shorter time span.

We also implemented simulated annealing and tabu search algorithm. To implement these algorithms we used an operator that switches a book was not scanned for a book that a signed up library scanning. After implementing these algorithms we noticed that were no improvements to our solution.