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

Papers Consulted:

- Computing the Initial Temperature of Simulated Annealing
- Crossover Operators
- Comparative Study of Various Cooling Schedules for Location Area Planning in Cellular Networks Using Simulated Annealing
- Dynamic Simulated Annealing for solving the Traveling Salesman Problem with Cooling Enhancer and Modified Acceptance Probability
- A new crossover operator based on the rough set theory for genetic algorithms

Problem Formulation

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

Decision Variables: List of libraries and respective books

Neighborhood / Mutation: Swap the position of two libraries or books

Genetic algorithm: Two point (OX1)/single point crossover using the libraries from both parents

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

Evaluation: Sum of the scores of all books from libraries that can be signed in until the deadline

Implemented Work ~ Input & Output

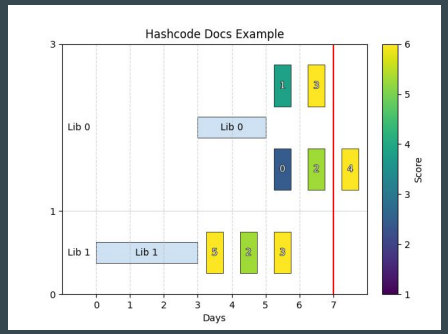
To obtain our solution we are using Python and pypy3. Python is used when we want to use our visualizer, otherwise pypy3 returns a solution in less than half the time. Our main data structure is a class (“Problem”).

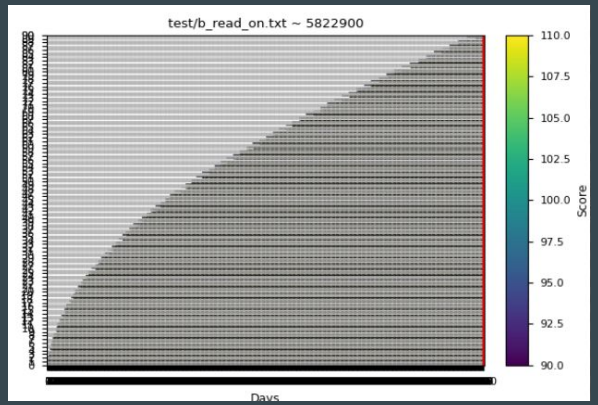
The first line of the input file contains three different integers that represent the number of different books, the number of libraries and the number of days in this order.

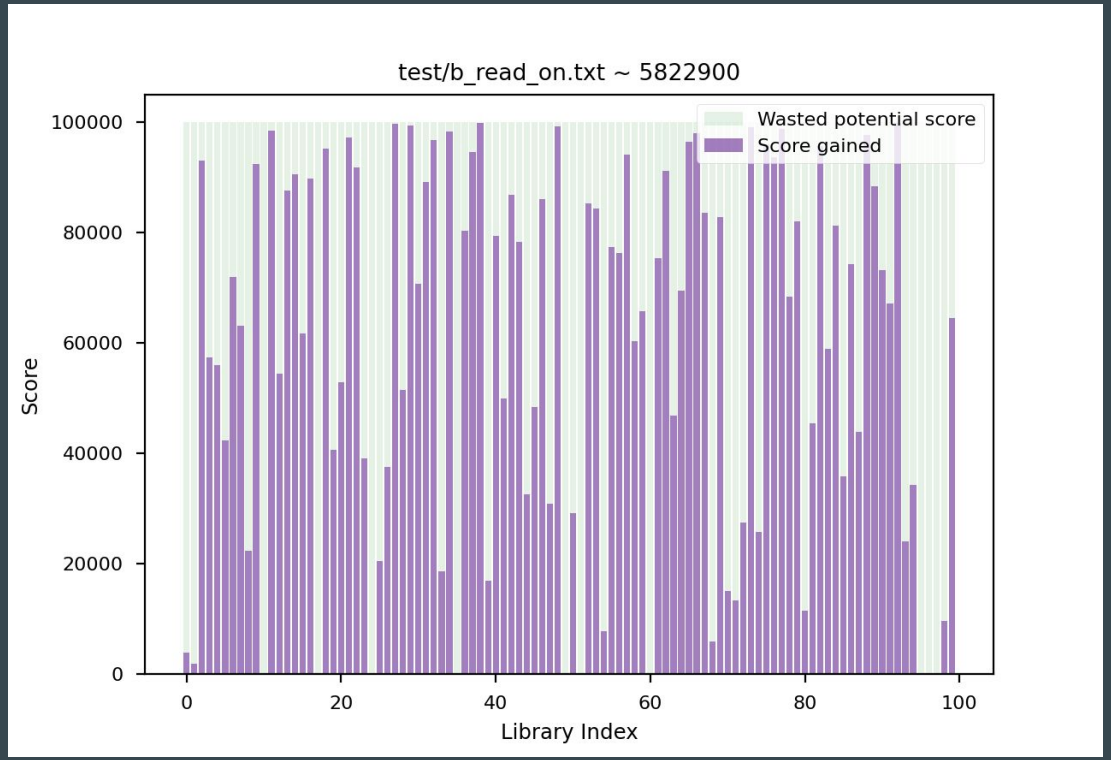
This is followed by L sections describing individual libraries from library 0 to library L–1. Each section contains two lines: the first one contains the number of books in the library, followed by the number of days it takes to sign in the library ending with the number of books that can be shipped 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 (one works the other was deleted because of performance issues).

Solution Visualization ~ Current Version on the right







Implemented Work

We have implemented a parser for the input files and a way to output the solution to an output file. This output process also calculates the real maximized score of a solution. We have also implemented the First Choice Hill Climbing algorithm. This algorithm is very slow since calculating the large neighborhood is costly.

We also implemented simulated annealing with a dynamic cooling schedule. For this algorithm we implemented two operators, one to swap libraries and one to swap books. During each loop one of these is applied with a 50% chance. Our group considered this algorithm the best.

Lastly, we implemented the genetic algorithm using an elitist selection. After testing some crossover functions, we decided to define the single point crossover operator as default, as we found it to obtain better results within a respectable amount of time. The other operator implemented (two point crossover OX1) had worse results within the same timeframe. For this algorithm we generate a custom amount of starting solutions and apply the crossover function to them. On top of that, mutations can occur with 2.5% chance, that cause a library swap or a book swap.

Approach

To evaluate our solution a list containing all libraries is iterated until the deadline is reached. For each of the libraries in that time frame their books are iterated and until the deadline is reached the book's value is added unless that book is a duplicate that already has been scanned, in which case it has no value. This evaluation can be improved through a careful analysis of the arrays, however this comes with higher execution times, so we applied this final correct evaluation exclusively to the final solution.

The operator we are using for the genetic algorithm is a simple single point crossover. We choose the better half of the population and, for each one, we choose a second random parent. To generate the new child to replace one of the worst solutions we cut one of the parents in half and copy that to the child. The remaining libraries are chosen in the order of the other parent.

We also implemented a two point crossover (OX1), in which we select a sequence of libraries to maintain from a parent, and the remaining empty indexes are filled with the libraries from the other parent, in order of their appearance.

Algorithms implemented

Hill Climbing First-Choice:

- This algorithm has poor performance, since iterating and evaluating the neighborhood is slow (large neighborhood size)
- We are scanning the minimum number of neighbors possible (no repetition)
- Does not include the operator to swap books in the neighborhood for performance reasons (to ensure it can be completed in useful time) since during all our tests it had no impact on the final scores

Hill Climbing Steepest Ascent:

- Auxiliar function obtains all neighbors
- Does not compute a solution in effective time

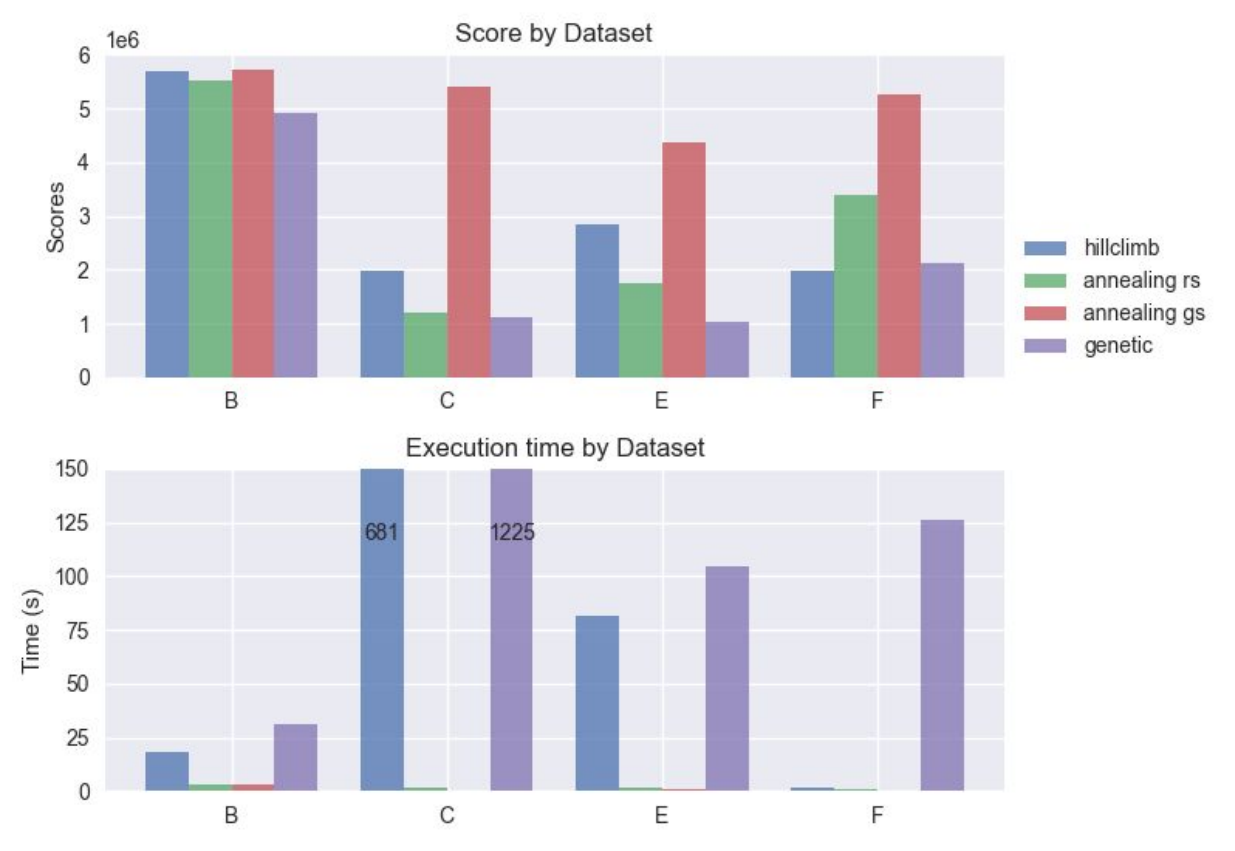
Simulated Annealing:

- Custom starting Temperature as well as Initial & Final Cooling
- Dynamic Cooling Schedule
- Two operators (swap books and libraries)
- 50% chance of performing either a library swap or book swap operation

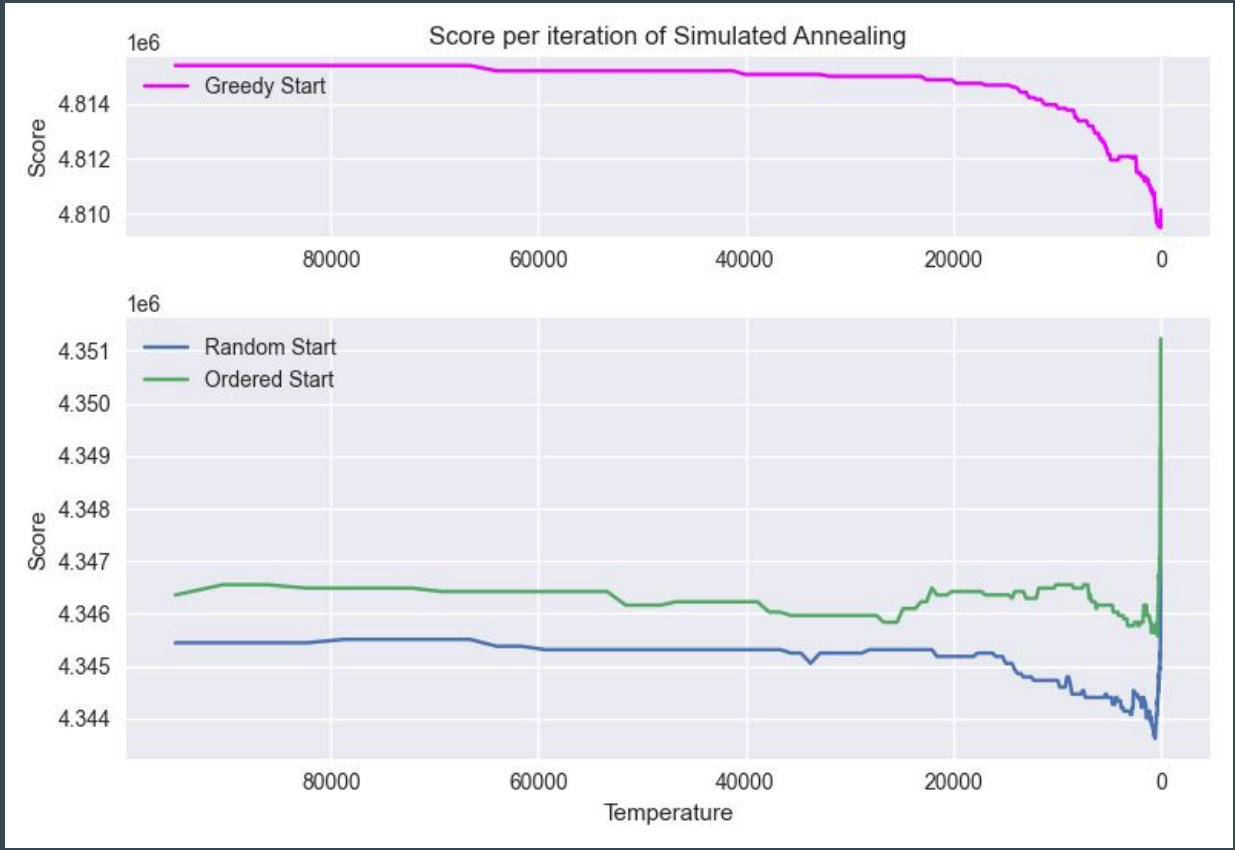
Genetic Algorithm:

- Custom population size, max generations, crossover operator and mutation chance
- Single point and two point crossover (OX1) using libraries
- Mutations occur with 2.5% chance swapping books or libraries
- Elitist Selection

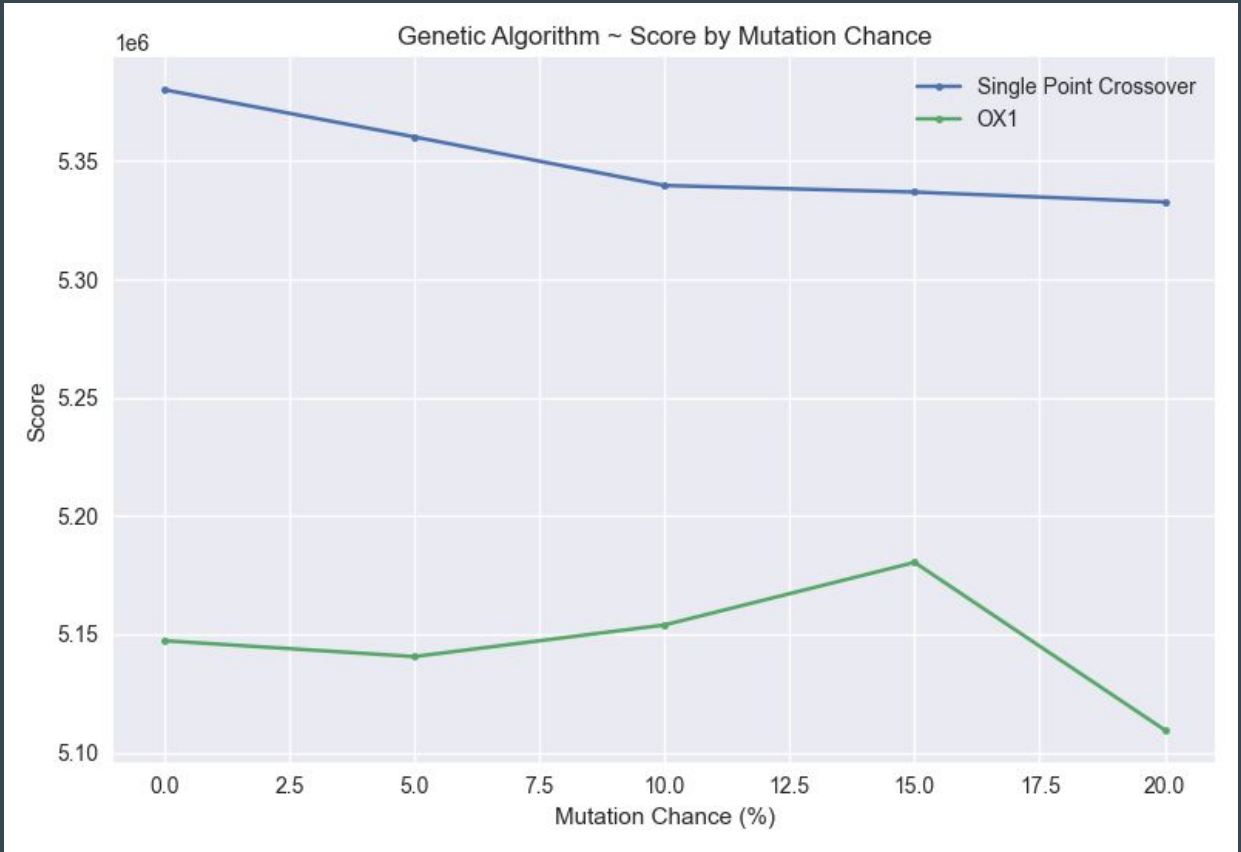
Experimental results



Experimental results



Experimental results



Conclusions

When dealing with a many different book values, our implementation of the algorithms generates sub-optimal results. However, when dealing with similar book values and many libraries our implementation creates good results.

Overall we noticed that the simulated annealing algorithm obtains the bests results while taking the shortest computation time. We implemented a dynamic cooling schedule, where the cooling rate is linearly interpolated between two values according to the Temperature, allowing more iterations with a lower temperature which ensures we can better explore the neighborhood before returning the solution.

We also noticed that when trying to optimize a solution obtained by a greedy algorithm, our algorithms rarely managed to optimize it. As such, we decided to allow the starting solution to be customized as an additional flag (by default it randomly shuffles the libraries and books).

Finally, since exploring and evaluating the entire neighborhood is too costly, we decided to not delve any further into both the steepest ascent and tabu search algorithms.