POLITECHIKA POZNAŃSKA

Combinatorial Optimization Project Report

Greedy + Genetic Algorithm Solution

Mateusz Tabaszewski 151945 Group III

2022/2023

Introduction and Problem Description

The Program was compiled for the sake of tests on Ubuntu operating system, using the command:

gcc –O2 –g0 Program.cpp –o Program.exe –lstdc++ -lm

It is advised to compile the program using the same flags. The program was tested on an I5-7200U (7^{th} generation, 2 cores), 2.5 GHz, 8 GB RAM computer and it managed to finish all calculations within the specified time frame.

This Report is meant to be an explanation and description of a C++ program solving an NP-Hard problem written for Combinatorial Optimization classes. The paper is meant to be thorough description of steps taken to solve the problem. The report includes description of methodology, different approaches taken, their results, implementation details as well as conclusions drawn from the experiments. The problem in consideration was taken from Google Hashcode competition preliminaries from 2020. All algorithms presented in the report have been judged according to 2 criteria-time necessary to find a solution as well as number of points scored on each file. For the purposes of testing the scores reached by an algorithm a testing program from an eKursy website was used. Additionally the time requirement dictated that the algorithm should find a solution for a single file in time no longer than 5 minutes.

The problem to be solved belonged to the class of NP-Hard problems. This particular challenged was originally presented during the Google Hashcode competition preliminaries. Shortened description of the problem is as follows: There are **B** different books each one with a score $\mathbf{S}_i$ gained upon scanning a book. Additionally, there are **L** different libraries, with each library being represented by a set of books present in their library, with the size of the set represented by $\mathbf{N}_j$, $\mathbf{T}_j$-number of days it takes for each library to sign up and $\mathbf{M}_j$ – number of books per day that can be sent by
a library after completing the sing-up process. Lastly a deadline is introduced, represented by a variable **D**-showing the number of days when the sign-up and scanning actions can be taken. First the algorithm is meant to pick a library to be signed up, only one library can take part in a sing-up process at a time and once it is finished, the algorithm is meant to select books to be scanned. Each book offers a certain score but no score is given upon scanning a duplicate. The library will continue scanning $\mathbf{M}_j$ books a day until the deadline or until all books in the library have been scanned. The libraries can continue scanning books concurrently and even when a sign-up process is taking place for a different library. The goal of the algorithm is to prioritize libraries and books in such a way as to maximize the number of points gained upon reaching the deadline.

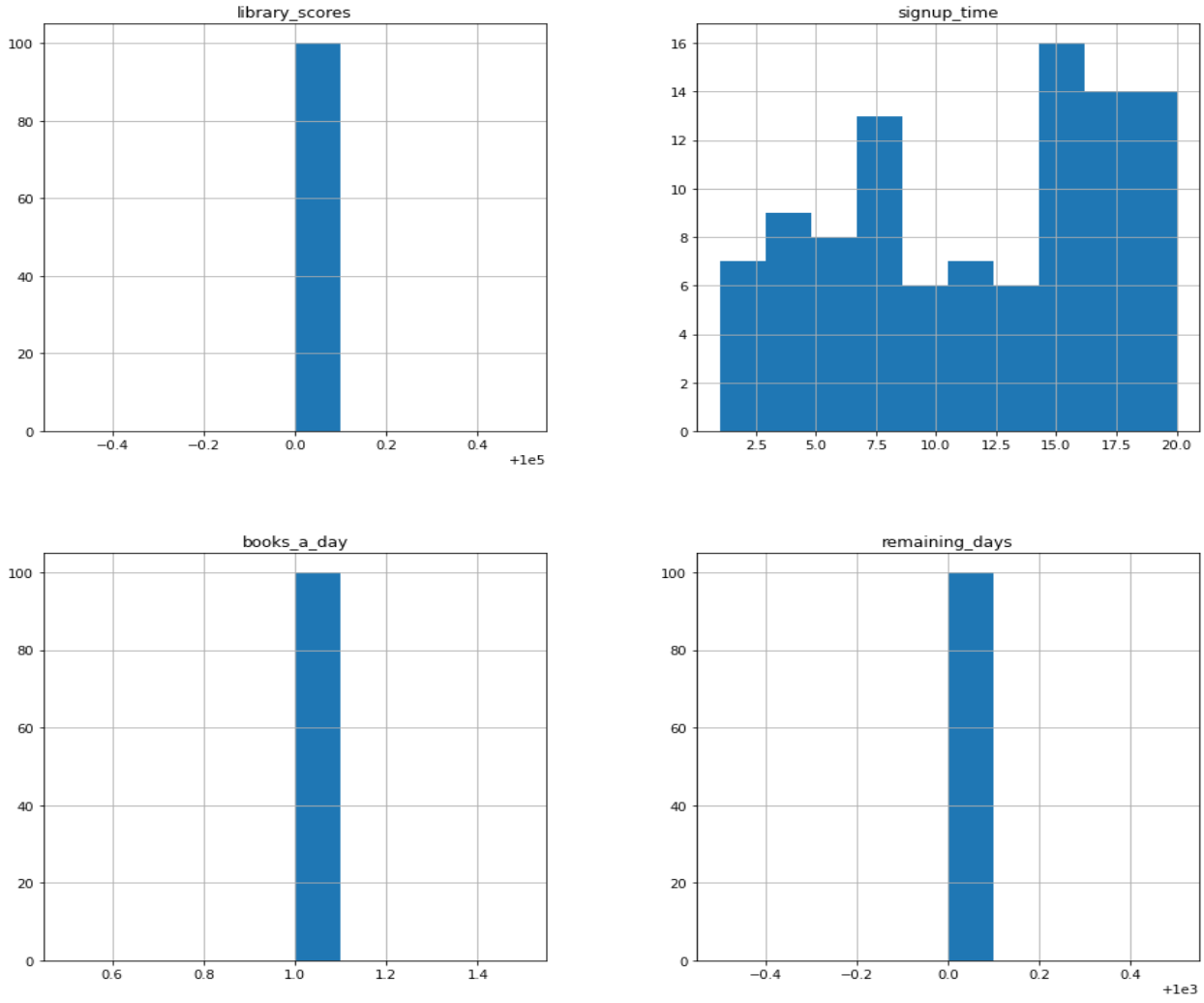
It is important to note that the previously mentioned variables have the following constraints:

- $1 \leq \mathbf{B} \leq 10^5$
- $1 \leq \mathbf{L} \leq 10^5$
- $1 \leq \mathbf{D} \leq 10^5$
- $1 \leq \mathbf{N}_j \leq 10^5$
- $1 \leq \mathbf{M}_j \leq 10^5$
- $1 \leq \mathbf{T}_j \leq 10^5$
- $1 \leq \mathbf{S}_i \leq 10^3$

Data Analysis

In order to better understand the given file data and the given problem, a brief data analysis was conducted. For this purpose Python and its pandas library was used. The analysis also gave insight into what files would later turn out to be most problematic for the algorithm. Vast majority of the analysis concentrated around visualizing the data using histograms and gaining information about the basic measures of variability through the .describe() method. Data analysis for the file "a_example" was omitted due to the fact that the file's small size and minimal score book values meant that it would not meaningfully affect the final score, also finding the optimal score for a was also relatively easy so there was no need to analyze this data further.

- Analysis for file "b_ read_on":



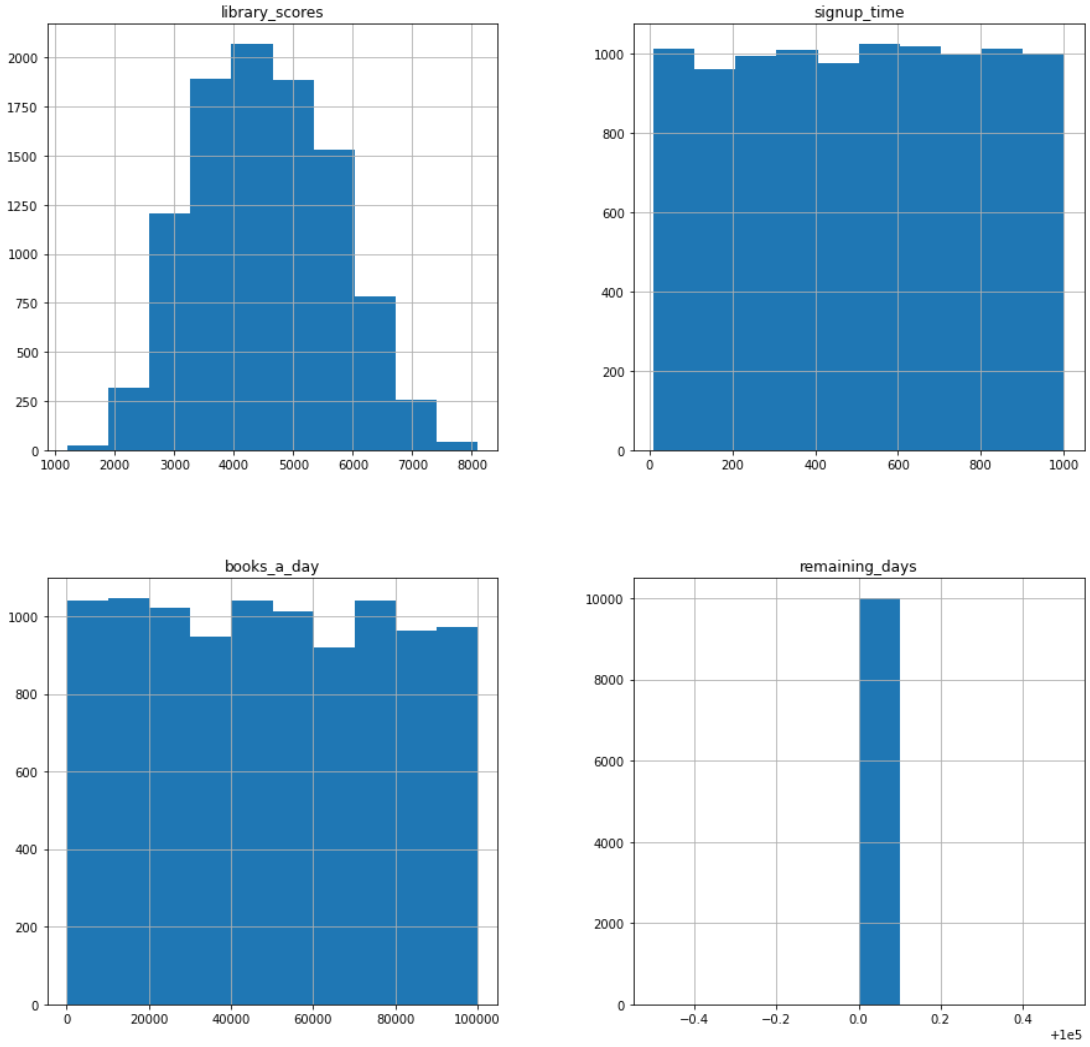
Figures 1-4. The graphs are all histograms, showing scores of libraries, number of books that can be sent in a day and number of remaining days(this will always be a single bin for all files since it is a feature independent of libraries and is on the graph only as a byproduct of using the .hist() method on a pandas data frame).

Table 1. Showing basic summary for data in file b.

	library_scores	signup_time	books_a_day	remaining_days
count	100.0	100.00000	100.0	100.0
mean	100000.0	11.81000	1.0	1000.0
std	0.0	5.95046	0.0	0.0
min	100000.0	1.00000	1.0	1000.0
25%	100000.0	7.00000	1.0	1000.0
50%	100000.0	12.50000	1.0	1000.0
75%	100000.0	17.00000	1.0	1000.0
max	100000.0	20.00000	1.0	1000.0

This file contains libraries with the exact same score and the same rate of books sent a day. The only difference between libraries being the necessary signup time. As such, this file should be relatively easy to find a solution close to the optimum since the variability of data is very small and the best way to success with this file is to prioritize files with smallest signup times.

- Analysis for file “c_ incunabula”:



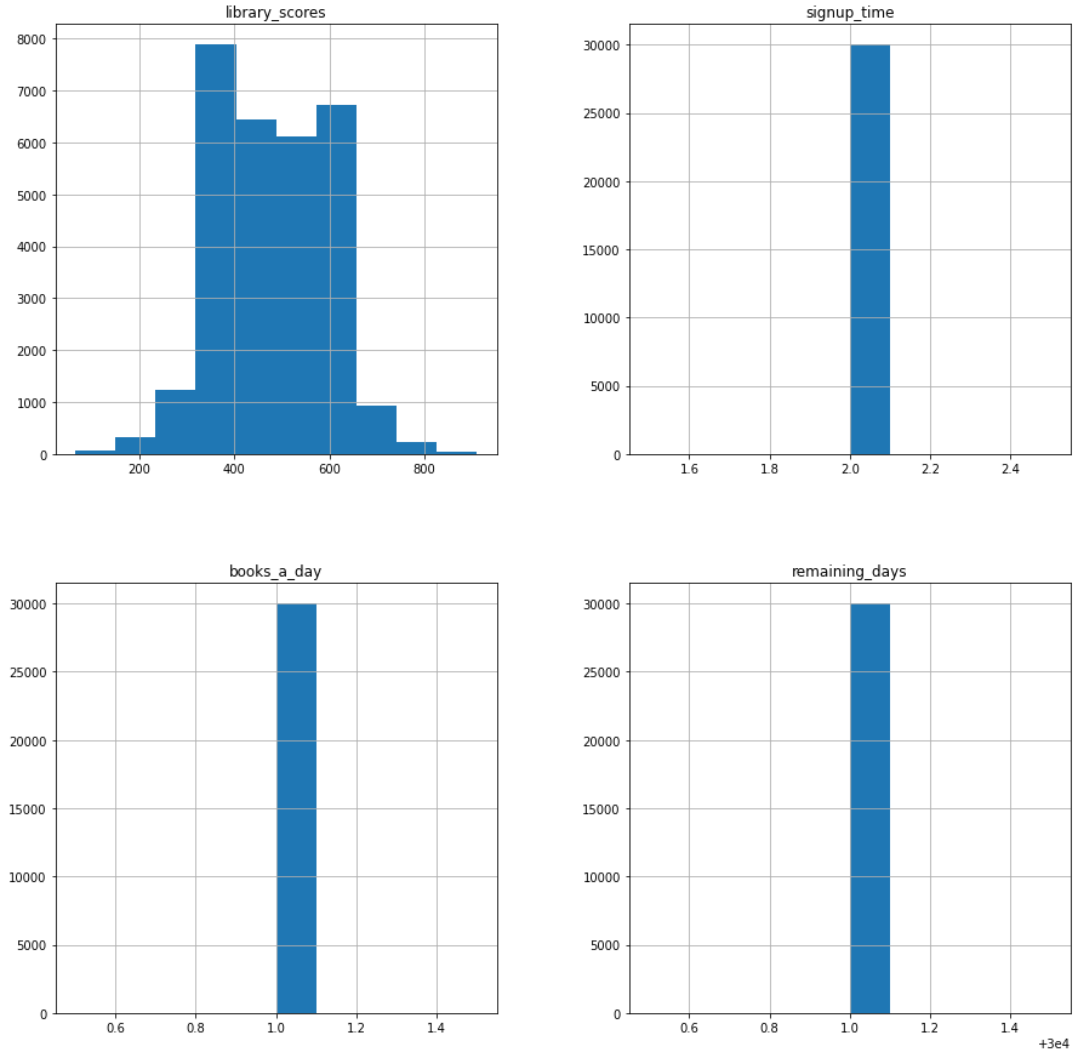
Figures 5-8. The histograms show library scores, signup time, number of books a library can send a day and number of remaining days respectively.

Table 2. Table with a basic summary of values present in the file c.

	library_scores	signup_time	books_a_day	remaining_days
count	10000.000000	10000.000000	10000.00000	10000.0
mean	4520.910000	506.079700	49511.37750	100000.0
std	1160.414013	285.962408	28887.29776	0.0
min	1206.000000	10.000000	210.00000	100000.0
25%	3622.750000	260.000000	24005.00000	100000.0
50%	4471.000000	510.000000	49334.00000	100000.0
75%	5374.000000	753.250000	74783.00000	100000.0
max	8093.000000	1000.000000	99998.00000	100000.0

As can be seen the variability of data for the file c is much higher than of file b and it is not immediately clear what algorithm should be used to gain the best results. It is also worth noting that the library score graph shows approximately a normal distribution. This hints at a fact that it may be worth it to prioritize libraries with higher scores, however the specific implementation of that approach will be discussed later.

- Analysis of file “d_ tough_choices”:



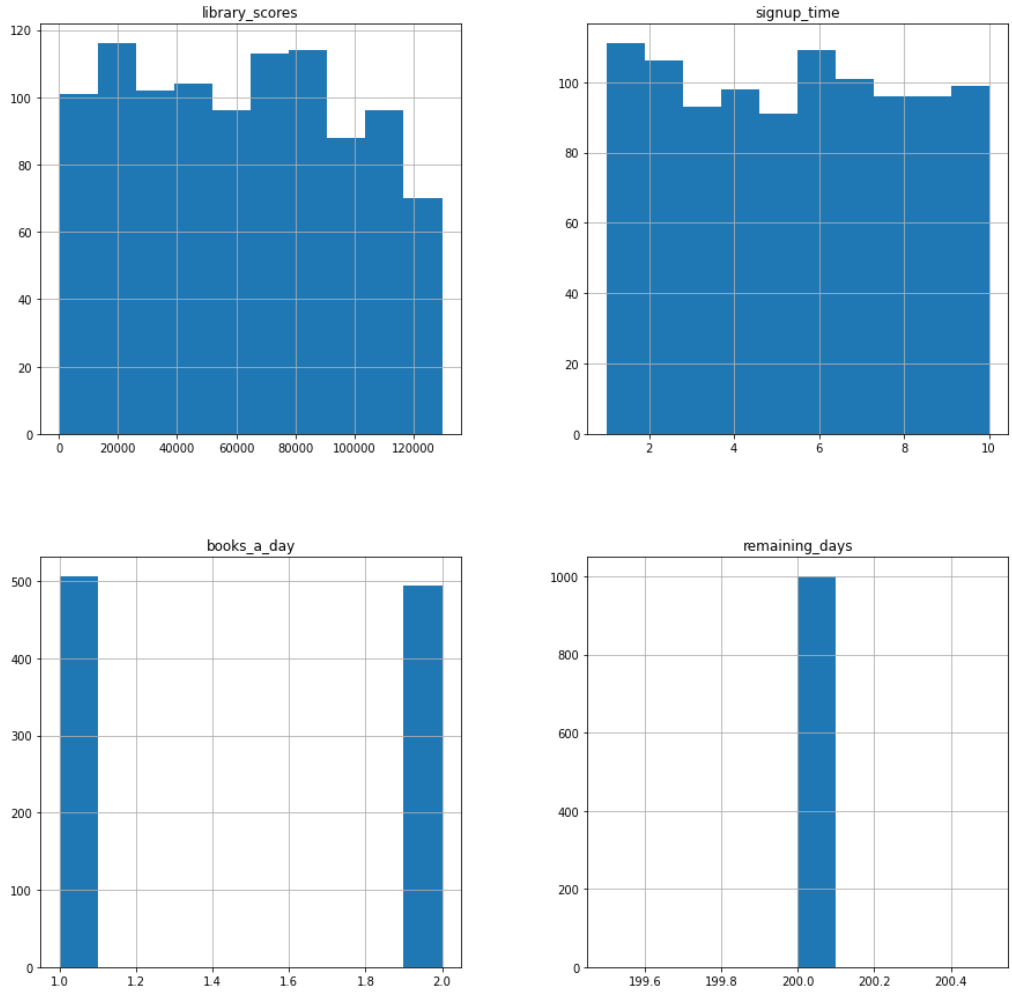
Figures 8-12. Histograms showing total library scores, signup time, number of books that can be shipped in a day and number of remaining days for the file d. The histograms were created based in data parsed with respect to each library.

Table 3. Table with a basic summary of data for file d.

	library_scores	signup_time	books_a_day	remaining_days
count	30000.000000	30000.0	30000.0	30000.0
mean	478.400000	2.0	1.0	30001.0
std	117.058511	0.0	0.0	0.0
min	65.000000	2.0	1.0	30001.0
25%	390.000000	2.0	1.0	30001.0
50%	455.000000	2.0	1.0	30001.0
75%	585.000000	2.0	1.0	30001.0
max	910.000000	2.0	1.0	30001.0

Data presented in the file d has much less variability than the one present in the file c. The only difference between libraries is the total score from books. The signup time and number of books that can be sent in a day are the same for all libraries. This means that this file should also be easier to solve than the c file. This file has the highest number of libraries and the longest simulation time (~30000 simulation days), which may increase the time it takes for the algorithm to solve this problem.

- Analysis for file “e_so_many_books”:



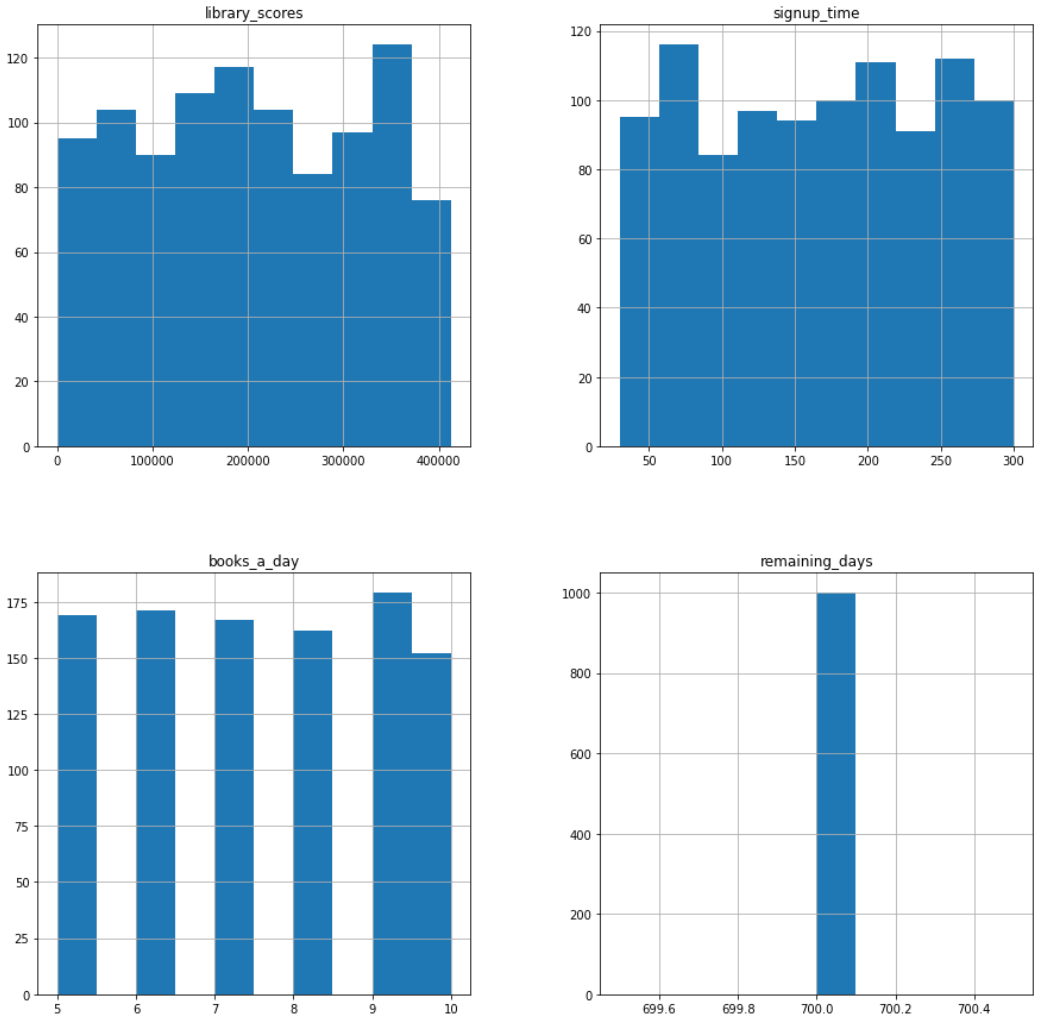
Figures 13-16. Histograms showing library scores, signup time, number of books that can be sent in a day and number of remaining days for the data of file e.

Table 4. Table with a basic summary of the data of file e.

	library_scores	signup_time	books_a_day	remaining_days
count	1000.000000	1000.000000	1000.000000	1000.0
mean	61690.074000	5.432000	1.494000	200.0
std	36087.861015	2.899963	0.500214	0.0
min	227.000000	1.000000	1.000000	200.0
25%	30909.750000	3.000000	1.000000	200.0
50%	61578.000000	6.000000	1.000000	200.0
75%	91328.000000	8.000000	2.000000	200.0
max	129651.000000	10.000000	2.000000	200.0

As can be seen on Fig. 13 and 14, the library scores and signup time for data presented in the file "e" do not have any easily noticeable distribution, also both variables can achieve a range of values meaning that deciding which libraries to prioritize is not immediately obvious. It also appears that libraries can only send either 1 or 2 books a day, narrowing the range of values possible for the $\mathbf{M}_j$ variable for the file e and hopefully simplifying the decision making process for the algorithm.

- Data Analysis for file f_ libraries_of_the_world:



Figures 17-20. Histograms showing library scores, signup time, number of books that can be sent in a day and number of remaining days for the data presented in file f.

Table 5. Table showing basic data summary for file f.

	library_scores	signup_time	books_a_day	remaining_days
count	1000.000000	1000.000000	1000.000000	1000.0
mean	204421.248000	166.192000	7.467000	700.0
std	116632.444949	78.715267	1.696405	0.0
min	164.000000	30.000000	5.000000	700.0
25%	107310.250000	95.750000	6.000000	700.0
50%	201436.500000	168.500000	7.000000	700.0
75%	310847.250000	235.000000	9.000000	700.0
max	412714.000000	300.000000	10.000000	700.0

As has been presented, the data of file f also takes a wide range of values similarly to data presented in the file c. Although, it seems that range of values for $\mathbf{M_j}$ is less than it was for the c file. Regardless, achieving optimal solution for file f may end up being more problematic than for other files(barring the c file).

In conclusion, files which require the most attention and which may end up being the hardest to find acceptable solutions to are files c, f and d (due to time limit) . Also it seems that taking into account library scores, signup time and number of books a library can send a day, may be beneficial towards building an algorithm that solves this problem. Of course libraries with higher scores should be prioritized, as should be those with shorter signup time and those that can send more books in a day. It may also be beneficial to take into account variability of data or frequency of occurrence of certain books.

Description of the Algorithm

The algorithm solving the presented NP-Hard problem is a combination of two different algorithms. First, it is important to realize that due to the time constraints the algorithm should be very time-efficient. On top of that the presented problem could be considered as a combination of problems, firstly sorting the libraries and secondly sorting the books in the library from the most desirable to the least. As such the first component of the algorithm was a Greedy Algorithm. This solution was chosen thanks to its time efficiency and a certain quality guarantee (which may not have been the case with algorithms involving random components like Local Search or Genetic Algorithms). Additionally, due to size of the data files, it may end up being inefficient and difficult to predict which solutions have the best long-term effects, as such it seems that the logical alternative involves creating an algorithm which picks solutions which appear the best at a given moment. The greedy algorithm picks libraries according to a scoring function, essentially making the best short-term decision based on score (this score denoted as $\mathbf{S_F}$) obtained from this function. However, it is not immediately clear how to judge which library and which order of books should be taken as the best immediate solution. As such, multiple tests have been conducted which have been presented in more detail in one of the later sections of this paper. In the final version of the algorithm, multiple formulas to obtain the score of the library have been used simultaneously, as such the program generates multiple solutions at the same time. The actual score ($\mathbf{S_{total}}$), i.e. the score of all scanned books across all the libraries, not counting duplicates, is then calculated for each solution. The solution with the highest $\mathbf{S_{total}}$ is remembered as the best solution thus far. After this first part is concluded a second phase of the algorithm starts.

Once solutions based on different measures of score $\mathbf{S_F}$ are all found, then the algorithm tries to search through other possible solutions based on already obtained solutions via Genetic Algorithm. First, the solutions obtained in the first phase are mutated in order to create initial population. There are 2 types of mutations present in the algorithm- **Greater mutation** and **lesser mutation**. Greater mutation changes the order of libraries inside a solution while lesser mutation changes the order of books inside a library. Of course, when the order of libraries is changed, the number of books that can be sent changes. As such, the algorithm has been designed in such a way that during fitness evaluation, only those books are taken into account that would be able to be shipped out (for more details go to the “Implementation Details” section). After the algorithm performs both mutations a certain number of times on each individual, a fitness function evaluates the solutions in such a way that the fitness of each solution is equal to $\mathbf{S_{total}}$ for that solution. If during this phase a solution better than the previously best one is found, then it becomes the new best solution and is remembered. Lastly, through tournament selection population to be mutated is selected. The process is repeated until time approaches a deadline within some uncertainty measures (as to not terminate the program too late).

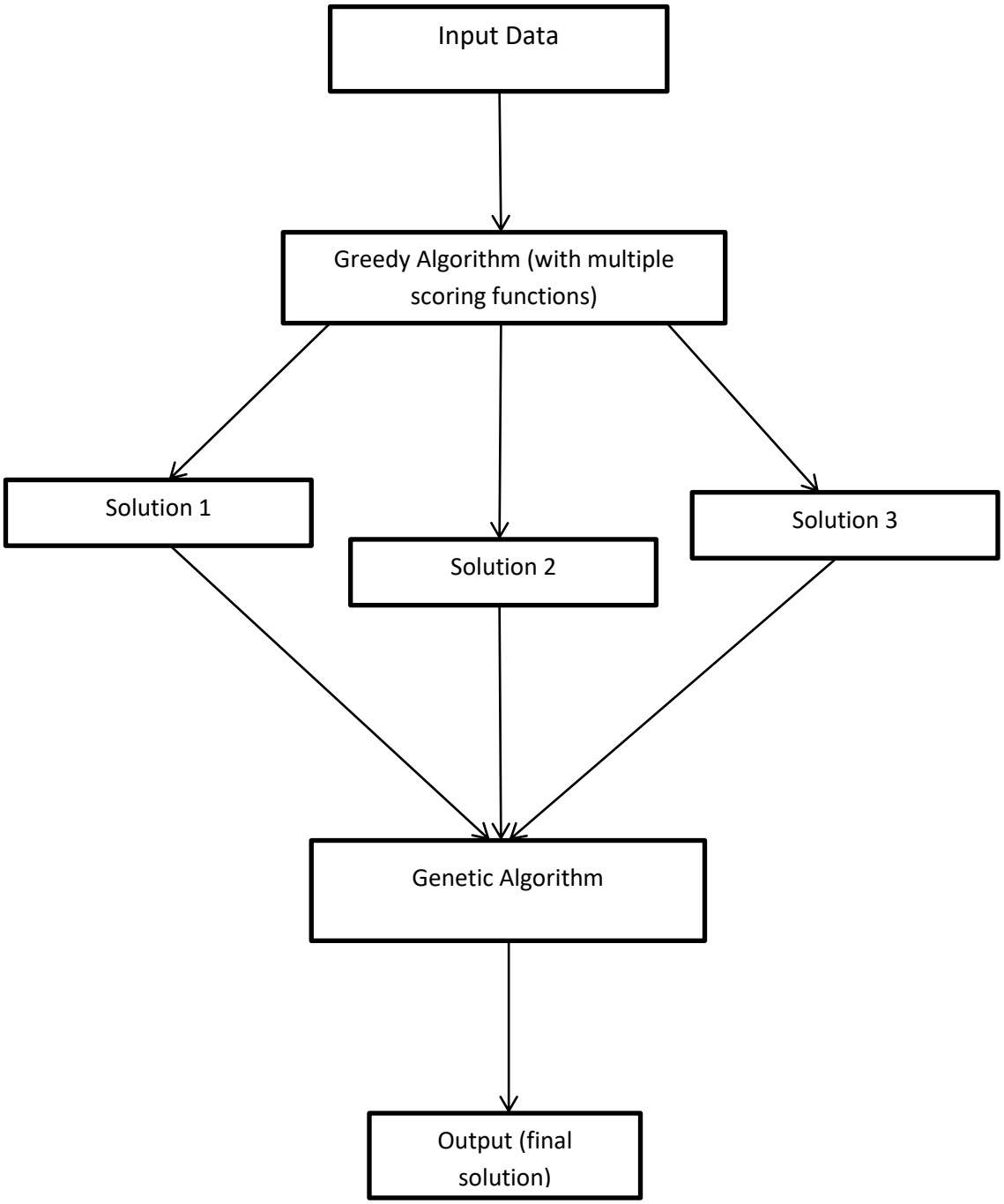


Figure 21. Simplified schematic of the algorithm solving the NP-Hard problem. Note: The final version of the greedy algorithm does not necessarily need to produce only 3 solutions, this is just a simplified visualization meant to get across a general idea of how the algorithm work, for details regarding number of solutions generated by the greedy part of the algorithm please refer to the "Implementation Details" section of the paper.

Analysis of the Greedy Algorithm

This section is concerned with the first part of the algorithm, i.e. the greedy algorithm. The figures and tables shown below represent different approaches towards calculating the score responsible for choosing a given library $\mathbf{S_F}$ and as such resulted in different $\mathbf{S_{total}}$. It is important to note that it is not always the case that $\sum \mathbf{S_F} = \mathbf{S_{total}}$. In fact, for the majority of the considered below formulas $\sum \mathbf{S_F} \neq \mathbf{S_{total}}$. It is the case because $\mathbf{S_F}$ is a score given to every library by some arbitrary evaluation function in every iteration of the loop (it is calculated again for the libraries that have not been chosen after updating the scores of books to prevent selection of duplicates). $\mathbf{S_F}$ only determines which library should be chosen while $\mathbf{S_{total}}$ determines actual score of all scanned books based on which the algorithm is evaluated. However, from the considered formulas, there is a single case for which $\sum \mathbf{S_F} = \mathbf{S_{total}}$.

- Equation number 1:

$$S_F = \sum_{0}^{\inf((D_c - T_j) * M_j, N_j)} S_k$$

where k is meant to index books present in the j^{th} library from those with highest scores to those with the lowest scores, in such a way that $S_0 >= S_1 >= S_2 >= \ldots. >= S_n$. S_k was written instead of S_i, since i indexes books based on the way data was initially presented, which is not necessarily a sorted order required for this algorithm. The algorithm updates books' scores after every main loop iteration. This update only changes (assigns 0) values for books that are assigned to be scanned from the picked library in order to force the algorithm to not pick the same books in the future when there is a possibility for a better book to be picked. D_c is meant to represent currently remaining number of days, where starting value of D_c is equal to D and after every iteration of the main loop D_c is decreased by T_j where j is the index of the chosen library. N_j represents number of books in a library.

Table 6. Table showing results(S_{total}) of the Greedy Algorithm using equation 1 for each file.

Equation 1			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	21	100%
B	10000000	5822900	58.23%
C	30076415	1480105	4.92%
D	5109000	5028530	98.42%
E	12548648	3323526	26.49%
F	40111142	1832987	4.57%
Total	97845226	17488069	17.87%

For this equation $\sum \mathbf{S_F}=\mathbf{S}_{total}$ as the formula essentially calculates how much score will be gained from a library by taking into account the number of remaining days until the deadline(Dc), the number of books a library can send in a day as well as the score of each book. This means that the program will calculate the exact number of points which will be gained by each library and picks the one that seems to be the best at the moment. After that book scores are updated. This approach has seems to have achieved promising results overall, however the scores gained for files c and f are comparatively very low and as such further trials should aim to eliminate this large error margin present for the aforementioned files.

- Equation number 2:

$$S_F = \frac{\sum_0^{\inf((D_c - T_j)*M_j, N_j)} S_k}{T_j}$$

The meaning for the variables present in the equation is analogous to the one in Equation number 1.

Table 7. Table showing results of the Greedy Algorithm using equation 2 for each file. For percentage score green means improvement and red means worse score relative to the best score thus far.

Equation 2			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	21	100%
B	10000000	5822900	58.23%
C	30076415	5688727	18.91%
D	5109000	5028530	98.42%
E	12548648	4986276	39.74%
F	40111142	5308034	13.23%
Total	97845226	26834488	27.43%

This formula calculates how much score signing up with a given library will yield (like in equation number 1) and then divides that score by the sign up time for each library. Such technique was meant to “punish” libraries that take too long to sign-up. This algorithm significantly improved the total score and significantly improved the score gain for files c and f. As such, a lot of the future score functions for this problem will be based on this equation.

- Equation number 3:

$$S_F = \frac{\sum_0^{\inf((D_c - T_j)*M_j, N_j)} \frac{S_k}{f_k}}{T_j}$$

The meaning behind variables is the same as for the equation above with the addition of f_k which is meant to represent frequency of occurrence of books across all libraries.

Table 8. This table shows score results for equation 3.

Equation 3			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	21	100%
B	10000000	5822900	58.23%
C	30076415	5642437	18.76%
D	5109000	5028530	98.42%
E	12548648	4976696	39.66%
F	40111142	5282940	13.17%
Total	97845226	26753524	27.34%

This formula is fairy similar to those previously mentioned with the addition that score of each book is divided by its frequency of occurrence among all libraries. The idea is to "punish" libraries for having too many frequently occurring books. However, the total score S_{total} is lower than for equation 2 and there is no improvement for any of the considered files.

- Equation number 4:

$$S_F = \frac{\sum_0^{\inf((D_c - T_j) * M_j, N_j)} S_k}{T_j * v_j}$$

where:

$$v_j = \frac{std(S_k)}{\overline{S_j}}$$

where:

$$\overline{S_j} = \frac{\sum_0^{N_j} S_k}{N_j}$$

This equation includes variable v_j which is meant to symbolize the variability index for the j^{th} library. The variability index is calculated as standard deviation of scores of a library divided by mean of scores of the same library.

Table 9. Table showing results of a Greedy Algorithm for equation number 4.

Equation 4			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	17	80.95%
B	10000000	5822900	58.23%
C	30076415	5499043	18.28%
D	5109000	4991025	97.69%
E	12548648	3782037	30.14%
F	40111142	3257117	8.12%
Total	97845226	23352139	23.87%

The idea behind this formula is to "punish" libraries with high variability of book scores. However, the scores for almost all files worsened and as such this approach does not seem viable. As such, it may be possible that the opposite approach may be more beneficial.

- Equation number 5:

$$S_F = \frac{\sum_0^{\inf((D_c - T_j) * M_j, N_j)} S_k}{T_j} * v_j$$

where:

$$v_j = \frac{std(S_k)}{\overline{S_j}}$$

where:

$$\overline{S_j} = \frac{\sum_0^{N_j} S_k}{N_j}$$

The meaning behind variables is the same as for the equation number 5.

Table 10. Table shows scores for Greedy Algorithm using scoring function number 5.

Equation 5			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	21	100.00%
B	10000000	5820600	58.21%
C	30076415	5560547	18.49%
D	5109000	4971330	97.31%
E	12548648	3497830	27.87%
F	40111142	5217797	13.01%
Total	97845226	25068125	25.62%

This equation was meant to encourage the algorithm to pick books with high data variability. However, taking variability index into account seems to have not yielded any improvements with regards to both total score and individual scores of each file.

- Equation number 6:

$$S_F = \frac{\sum_0^{\inf((D_c - T_j) * M_j, N_j)} S_k}{T_j} * std(S_k)^2$$

For this formula the variables are denoted in the same way as for the previous equations with the addition that $std(S_k)^2$ denotes variance of scores of books found in the j^{th} library.

Table 11. Showing results of scores gained by the Greedy Algorithm for equation number 6.

Equation 6			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	21	100.00%
B	10000000	5820600	58.21%
C	30076415	5629772	18.72%
D	5109000	5030155	98.46%
E	12548648	4131438	32.92%
F	40111142	5306535	13.23%
Total	97845226	25918521	26.49%

This equation is meant to prefer libraries with variability of data. As can be seen this did not improve the total score, however improvement for the file D can be seen. As such this equation will not be used on its own however it may be one of the number of equations used in the final version since it does provide improvement for at least one of the tested files.

- Equation number 7:

$$S_F = \frac{(\sum_0^{\inf((D_c - T_j) * M_j, N_j)} S_k) - n * \frac{e}{2}}{T_j}$$

The meaning for variables present in this equation is the same as for the previous equations with the addition of variables n and e. n is meant to be the number of occurrences of the bottom 25% of books according to their scores (among all libraries and not counting books with score already equal to 0, i.e. duplicates). On the other hand, e is meant to denote a score of the bottom 25% of books.

Table 12. Table showing results of scores gained by the Greedy Algorithm for equation number 7.

Equation 7			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	21	100.00%
B	10000000	5822900	58.23%
C	30076415	5686076	18.91%
D	5109000	5028530	98.42%
E	12548648	5002651	39.87%
F	40111142	5308034	13.23%
Total	97845226	26848212	27.44%

This formula has managed to improve the overall score as well as individual score for file “e”. This means that this equation has become one of the candidates to be used in the final version of the algorithm.

- Equation number 8:

$$S_F = \frac{(\sum_0^{\inf((D_c - T_j) * M_j, N_j)} S_k) - n * \frac{e}{2}}{T_j}$$

This formula is the exact same as equation number 7 with the exception that this time the threshold has been increased to the bottom 50% which means that n denotes number of books in a library that belong to the bottom 50% among all books. In a similar way e denotes total score of the bottom 50% of books from all libraries.

Table 13. Table showing results of scores gained by the Greedy Algorithm using formula number 8.

Equation 8			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	21	100.00%
B	10000000	5822900	58.23%
C	30076415	5645675	18.77%
D	5109000	5028530	98.42%
E	12548648	5041154	40.17%
F	40111142	5314266	13.25%
Total	97845226	26852546	27.44%

By increasing the threshold, the total score achieved by the algorithm improved further, scores also improved for files e and f. This equation has, as such, become the most likely candidate to be selected as one of the formulas to be used in the final version of the algorithm.

- Equation number 9

$$S_F = \frac{\sum_0^{\inf((D_c - T_j) * M_j, N_j)} S_k}{\log(T_j)}$$

This equation is exactly the same as formula number 2 with the only difference being that the score is now divided by $\log(T_j)$ instead of just being divided by T_j.

Table 14. Table showing results of scores gained by the Greedy Algorithm using formula number 9.

Equation 9			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	21	100.00%
B	10000000	5822900	58.23%
C	30076415	4246002	14.12%
D	5109000	5028530	98.42%
E	12548648	4957041	39.50%
F	40111142	5023570	12.52%
Total	97845226	25078064	25.63%

Unfortunately, the changes implemented in the equation number 9 did not yield any improvements.

- Equation number 10

$$S_F = \frac{\sum_0^{\inf((D_c - T_j) * M_j, N_j)} S_k}{{T_j}^2}$$

This formula is similar to formulas number 2 and number 9 with the only difference being that scores for each library are now divided by squared value of time necessary for the library to sign up.

Table 15. Table showing results of scores gained by the Greedy Algorithm using formula number 10.

Equation 10			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	17	80.95%
B	10000000	5822900	58.23%
C	30076415	2640802	8.78%
D	5109000	5028530	98.42%
E	12548648	4865305	38.77%
F	40111142	5084748	12.68%
Total	97845226	23442302	23.96%

This equation much like the previous one did not improve any of the scores and in fact, the results ended up being worse than for equation number 9.

- Equation number 11

$$S_F = \frac{\sum_0^{\inf((D_c - T_j) * M_j, N_j)} S_k}{\sqrt{T_j}}$$

Similar to previous equations with the only exception being that score is divided by square root of the sign-up time.

Table 16. Table showing results of scores gained by the Greedy Algorithm using formula number 11.

Equation 11			
File name	**Upper Bound**	**Score**	**Percentage Score**
A	21	17	80.95%
B	10000000	5822900	58.23%
C	30076415	5561419	18.49%
D	5109000	5028530	98.42%
E	12548648	4851690	38.66%
F	40111142	5340296	13.31%
Total	97845226	26604856	27.19%

As can be seen, this formula did not improve the overall total score across all files but it did provide improvement for file f. As such it can end up being one of the candidates for the use in final version of the Greedy Algorithm.

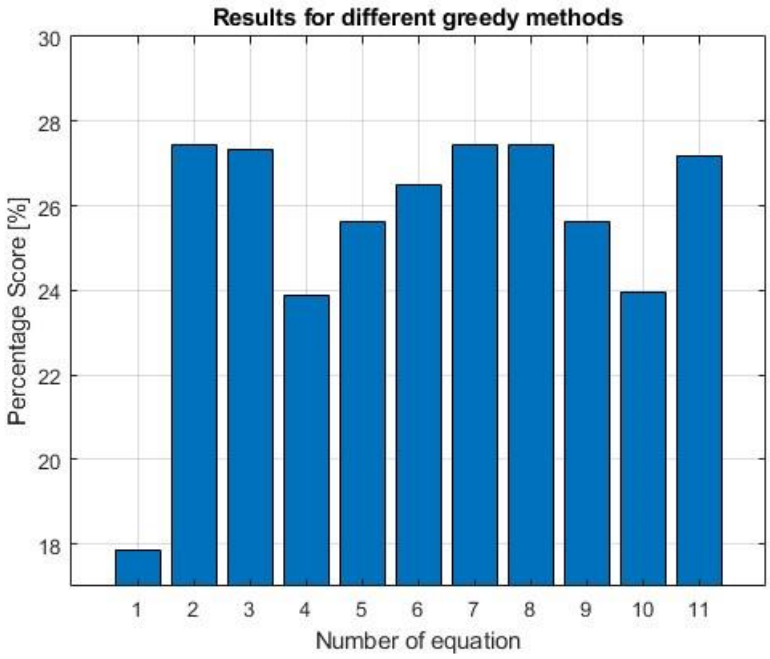


Figure 22. Graph showing percentage score comparison for all of the tested formulas. Note that the lowest value seen on the graph is not equal to 0, to more emphasize differences between achieved scores for different formulas.

- Conclusions:

Multiple scoring functions have been used and although many of them did not yield satisfying results some of them showed potential. When choosing the final scoring functions to be used in the final version of the algorithm two things have to be taken into consideration: calculation time and presence of improvement. Considering these factors fallowing factors have been chosen as candidates for the final solution:

-equation number 2

-equation number 6

-equation number 7

-equation number 8

-equation number 11

All of the above formulas provided some kind of an improvement, either in single or multiple files or in the total sum of scores of all libraries. None of them also require complex calculations that would significantly extend the time necessary for the algorithm to conclude. Equation number 2 is going to be the first formula to be selected since all remaining candidates are based on this equation, it was

also the first equation to improve the scores for all files. Equation number 8 is the second formula to be chosen as part of the final solution. This is due to the fact that it is preferred over equation number 7 thanks to its higher score and the idea behind both of these equations is similar. On top of that, equation number 8 has managed to gain the highest total score out of all the tested methods. Although the improvement may not be immediately obvious due to the scores being very close. Lastly, the equation number 11 was selected as it improves the score for the file which generally gained the lowest score-file f. In this case only 3 formulas were selected as it ensures that on most modern computer systems the program should have enough time to finish the calculations for all 3 variations of the Greedy Algorithm and start the next stage of the final algorithm-Genetic Algorithm.

In conclusion, multiple scoring functions for the Greedy Algorithm have been selected to maximize the chances of the algorithm to pick as close to optimal solutions no matter the complexity of the data in a file. All of the calculated solutions will be compared and variations of these solution will be selected as the starting population for the genetic algorithm.

Implementation Details

This section is meant to describe in greater detail the way the algorithm has been implemented. In more details this section should reiterate in greater detail the way the algorithm functions.

To reiterate the algorithm is composed of two sub-algorithms, the first is an implementation of a Greedy Algorithm with three different evaluations functions (as stated earlier the number of evaluation functions does not have to be equal to 3 but in this case it seems like the optimal value to allow the algorithm to fit within the time constraints on most modern computers for such big files). Once the Greedy Algorithm returns its solutions, different one for each evaluation function, the Genetic Algorithm is meant to create variations of these 3 solutions to fit the size requirements of the initial population. The Genetic Algorithm is meant to be performed until some arbitrary time threshold or max number of iteration are reached. From the previous paragraph based on score analysis for different files, the three chosen equations are: equation number 2, number 8, number 11. However, in order to avoid the confusing numbering of these equations from now on the equations will be referred to as equation α, β, γ respectively.

$$\alpha = \frac{\sum_0^{\inf((D_C - T_j)*M_j, N_j)} S_k}{T_j} \qquad \beta = \frac{(\sum_0^{\inf((D_C - T_j)*M_j, N_j)} S_k) - n*\frac{e}{2}}{T_j} \qquad \gamma = \frac{\sum_0^{\inf((D_C - T_j)*M_j, N_j)} S_k}{\sqrt{T_j}}$$

Explanation of meaning of the variables is present in the section “Analysis of the Greedy Algorithm”

- Before the main program body:

For the final version of the algorithm’s implementation libraries have been implemented in a form of a structure with containing number of books in a library, signup time, rate of books sent a

day, library score calculated in accordance with evaluation functions, index of the library, vector containing books and their scores and a variable “used” showing if the library has already been signed up. The program also contains a variety of global constants. Although not all of them will be described here the most important one is maxTime. When the total runtime of program for a single file exceeds the number of second specified by maxTime, the program will terminate. This allows the Genetic Algorithm to run longer for simpler files where the Greedy Algorithm finds solutions faster, this also allows better results to be achieved with a stronger computer (though it is not guaranteed due to the random nature of Genetic Algorithms).

- Greedy Algorithm

First the data is read and saved into the aforementioned structures with the caveat being the necessity to create copies for each library structure since the evaluation for the Greedy Algorithm is going to be performed separately for each α, β, γ respectively. Of course, this is more strenuous on memory but still negligible when compared to memory requirement for creating the Genetic Algorithm. In each library books are sorted in accordance with their scores. In this step counting sort algorithm is used, mostly because of its speed when compared to other algorithms. The time complexity of counting sort is O(n+k) with n being the number of elements and k being the range of elements. For this type of data counting sort is a great choice since n is at most equal to **B**($<=10^5$) and k is equal $\mathbf{S_i}$ ($<=10^3$) with both numbers being integers. This means that not only is counting sort possible but it is also viable thanks to small range of values relatively to the number of values. This means that the algorithm should perform the sorting operation very quickly. The sorting of the books is necessary to choose the most valuable books first whenever the sign-up is complete. On top of that, the libraries are meant to be judged by their evaluation function in accordance with what books they would be able to send in time. The operation of sorting is performed every time after a new library is joined and scores of books are updated in accordance with what books will already be sent from the chosen library. Because of that the sorting operation needs to be as quick as possible since the operation is performed a lot of times through the runtime of the algorithm. After the books are sorted, the algorithm calculates the values of libraries using the scoring function α while also taking into account the current best library score and if a library with score better than the current one, that is not yet part of the solution, is found, then it is remembered as the new best library. After calculating score for all libraries the best library is added to the solution. The same process is replicated for β and γ. This necessitates the existence of 3 separate solutions. At the end of the Greedy Algorithm, after all 3 solutions have been found, each of them is evaluated by calculating the sum of scores of scanned books in each of the signed-up libraries (of course duplicates give 0 points). The solution with the highest score out of these 3 is then remembered as the best solution.

- Genetic Algorithm

After obtaining 3 potential solutions and a current best solution from the Greedy Algorithm the program aims to find a better solution with the help of a Genetic Algorithm. First, all 3 solutions obtained in the Greedy Algorithm are copied without changes into the initial population, then in

order to fill the initial population, with initial population size=20, the solution obtained with evaluation α is then copied 6 times and before inserting it into the initial population it is mutated for each copy. The same is applied to solutions with evaluations β, γ with γ being replicated 5 times instead of 6 due to 17/3 not being an integer. When it comes to mutations in this algorithm, there are 2 separate types of mutations. First, whenever individual is meant to be mutated, there is some probability propOfLibsMutations. This probability specifies how likely it is for the first kind of mutation to occur in this individual. If the mutation succeeds a certain number of libraries is randomly selected and their order in the solution changes. The number of the libraries to be exchanged is randomly selected from 2 to some specified range. By changing the order of libraries in a solution, the number of books which can be sent on time also changes. However, this is later taken into consideration when calculating the total score/fitness of the solution. After the first mutation (no matter if it succeeded or not), the second kind of mutation is applied. This mutation has a certain probability propOfBooksMutations which specifies how likely it is for each library in the solution to have the book's order changed. This probability applies to each library in a solution and similarly to the previous mutation, the number of books to be exchanged is randomly selected from 2 to some specified range. After applying both kinds of mutations, a new individual is created. After filling the initial population, the algorithm calculates fitness of each solution as the score of books across all libraries in a solution that a library will have a time to scan after signing up. If there is a solution with better fitness than the current best solution, it becomes the new best solution. The selection process for the next population is implemented by using the tournament selection with the parameter k specifying the number of individuals considered in each tournament specified as: k=population_size/3.

Lastly, the program outputs the data in a way specified by the project assumptions.

Visualization of Results of the Final Algorithm

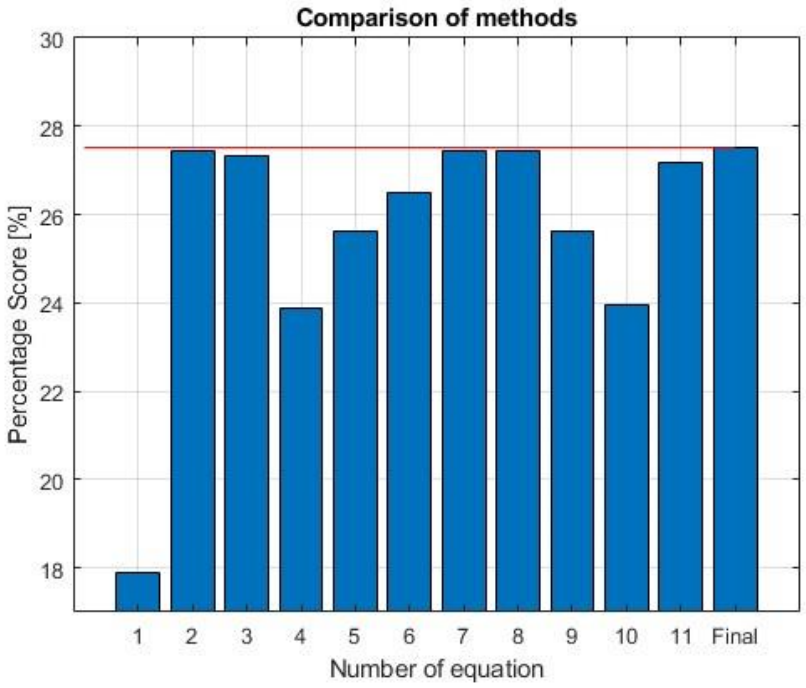


Figure 23. Graph showing comparison of the total scores achieved across all files depending on the formula used and the total score for the final version of the algorithm(Greedy + Genetic) denoted as “Final”.

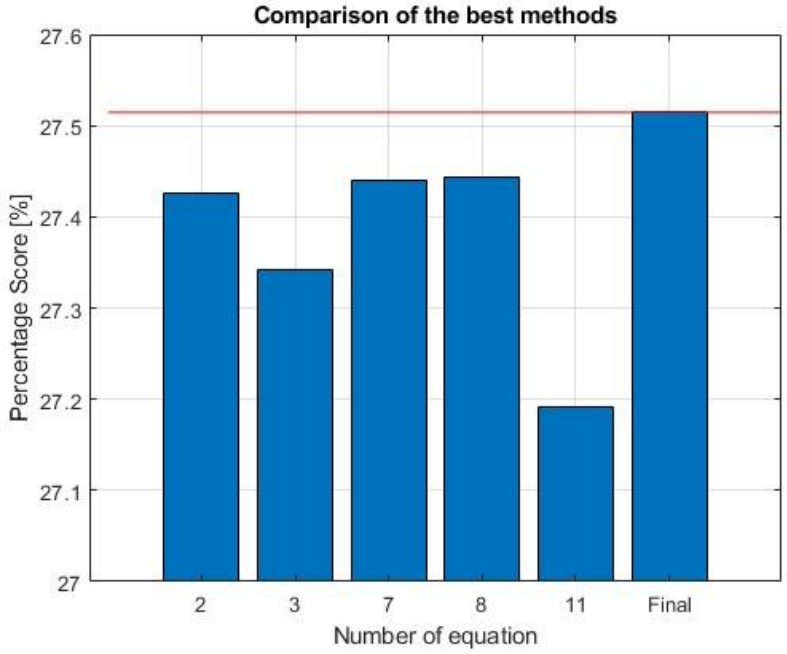


Figure 24. Graph showing the comparison of the total scores achieved across all files, for the algorithms using formulas which achieved the best results, compared to the score achieved by the final algorithm. The graph is scaled with the starting value=27%.

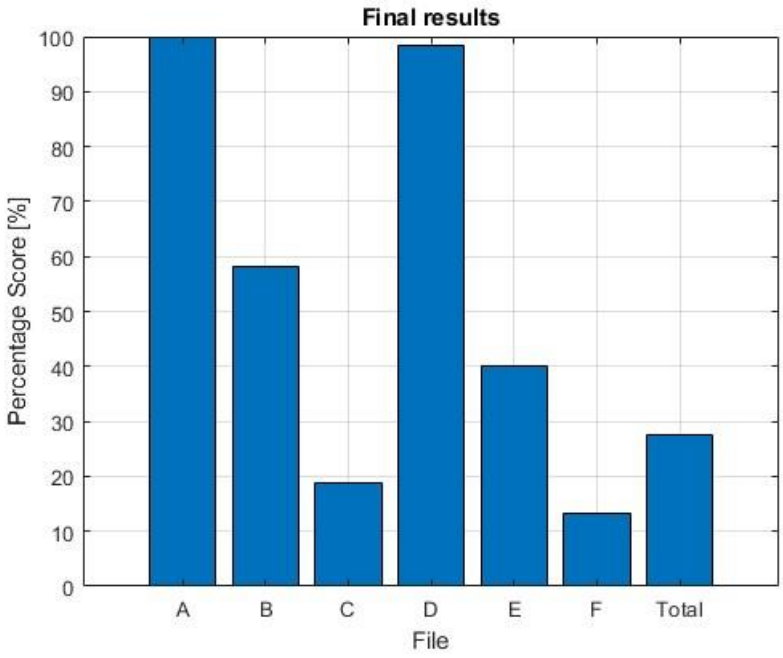


Figure 25. Graph showing the percentage results of the final algorithm for different files and the total score (percentages compared to the theoretical upper bound).

Conclusions

This project showed the difficulties associated with finding a time-efficient algorithm for solving an NP-Hard problem. For the future reference, multiple other methods could be applied in order to try to find a program which gives a solution as close to optimum as possible. In place of the Genetic Algorithm, Local Search techniques could have been applied including Simulated Annealing, with some limited neighborhood space of course. Similarly, it is possible that a better formula for the evaluation function could be found given enough time and perhaps more thorough data analysis with more emphasis put on regression techniques. However, given the challenge of solving an NP-Hard problem efficiently, I believe that this program gives satisfying results.

Literature

- J. Arabas, Wykłady z Algorytmów Ewolucyjnych, WNT Warszawa 2001
- Z. Michalewicz Algorytmy Genetyczne + Struktury Danych = Programy Ewolucyjne, WNT Warszawa 1996
- T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein Wprowadzenie do Algorytmów , PWN Warszawa 2012