Book Scanning Test Dataset

Book Scanning is a problem from the **Google Hash Code 2020** competition[1]. The task of the problem is: given a description of libraries and books available, plan which books to scan from which library to maximize the total score of all books, taking into account that each library needs to be signed up before it can ship books.

The Book Scanning problem is **NP-hard** because it combines multiple challenging optimization problems into one: a sequencing problem (determining the optimal order of library signups), a selection problem (choosing which books to scan from each library), and a resource allocation problem (optimizing within time constraints) - all with interdependencies where earlier decisions affect the value of later ones. It can be reduced from the set packing problem (selecting non-overlapping collections of books to maximize score) and job scheduling with deadlines (ordering library signups to maximize throughput before the deadline), both known NP-hard problems. This combination of optimization challenges with exponentially growing solution spaces means no polynomial-time algorithm can guarantee an optimal solution for all instances, requiring heuristic approaches for real-world applications.

There many approaches tried to solve this problem, where **greedy algorithms** with guiding simple heuristics are the most common[2] [3] [4] [5]. There are also some solutions that used **metaheuristic algorithms** to solve and optimize this problem[6] [7].

The **No Free Lunch Theorems**[8] establish that across all possible problem instances, no single optimization algorithm can outperform all others consistently. There are six instances for this problem given from Google Hash Code, where the first instance (*a_example.txt*) is just a toy instance and the other five are test instances[9]. In order to test the solutions of this problem more rigorously we have created more instances that cover the instance space in a better way, and using them we can do an **instance space analysis**[10] to find which algorithm works best for which set of instances based on their features.

---

[1] https://github.com/pierreavn/google-hashcode-archive/blob/main/archive/2020/qualification/hashcode_2020_qualification_round.pdf

[2] https://github.com/sagishporer/hashcode-2020-qualification/tree/master

[3] https://github.com/Kostero/ghc20/blob/master/kuba/optAss.cpp

[4] https://github.com/indjev99/Optimizational-Problems-Solutions/blob/master/book_scanning.cpp

[5] https://github.com/Nanored4498/Hash2020/tree/master

[6] https://github.com/pedromig/hashcode-models/tree/main/src/scanning

[7] https://github.com/eduardoCarneiro99/Google-2020-Hash-Code-Qualification-Round/tree/main

[8] https://homepages.uc.edu/~martinj/Philosophy%20and%20Religion/Arguments%20for%20the%20Existence%20of%20God/Teleology%20&%20Intelligent%20Design/Intelligent%20Design/Wolpert%20No%20Free%20Lunch.pdf

[9] https://github.com/pierreavn/google-hashcode-archive/tree/main/archive/2020/qualification/hashcode_2020_qualification_round.in

[10] https://dl.acm.org/doi/10.1145/3572895

The new test set comprises google hash code instances (5), real world instances (6) and synthetic instances (139).

Real world instances where generated using a web based tool[11] that gives the option to set the total number of books, total number of libraries, number of scanning days, minimum value for book score, maximum value for book score, minimum number of books per library, maximum number of books per library, minimum sign-up days per library, maximum sign-up days per library, minimum scanning rate of books per library and maximum scanning rate of books per library.

To generate the synthetic instances we first defined nine features for the instance:

- num_books: Total number of distinct books in the instance
- num_libraries: Total number of libraries available
- num_days: Total scanning days allocated
- average_book_score: Mean of all book scores
- variance_book_score: Variance of book score distribution
- books_per_library_avg: Average books per library collection
- signup_time_avg: Average library registration duration
- shippings_per_library_avg: Average daily shipping capacity
- book_duplication_rate: Ratio of books appearing in multiple libraries

Feature	Lower Bound	Upper Bound
num_books	1	100,000
num_libraries	1	100,000
num_days	1	100,000
average_book_score	0	1,000
variance_book_score	0	250,000
books_per_library_avg	1	100,000
signup_time_avg	1	100,000
shippings_per_library_avg	1	100,000
book_duplication_rate	0%	100%
book_scores	0	1,000
library_total_books	1	1,000,000
library_signup_days	1	100,000
library_books_per_day	1	100,000

Table 1. Value bounds

[11] https://bookscanning-ig.netlify.app/?authuser=0

The instance generation process employs evolutionary principles to create diverse instances. When multiple instances exist, the system predominantly (70% probability) performs crossover operations by selecting two parent instances and generating a new instance with feature values interpolated between them. Alternatively, when only one instance exists or with 30% probability otherwise, the system applies mutation by perturbing the features of a single parent instance. These derived features then determine the characteristics of the new instance. This approach creates a spectrum of problem instances with varying difficulty levels while maintaining structural similarities to their "parent" problems, effectively exploring the solution space and challenging different algorithmic strategies.

All the code used to generate synthetic instances can be found in the Book.Scanning.Dataset[12] repository. Using the scripts from this repository you can check the validity of the instances - meaning do they respect the bounds and limits of the problem, generate new instances, extract features from existing instances, reduce the dimensions of the instances using PCA (Principal Component Analysis) and plot the features in 2d.

In order to generate the plot that is shown in Figure 1 execute the scripts in this way:

1. scripts/extract_features.py
2. scripts/reduce_features.py
3. scripts/plot_features.py

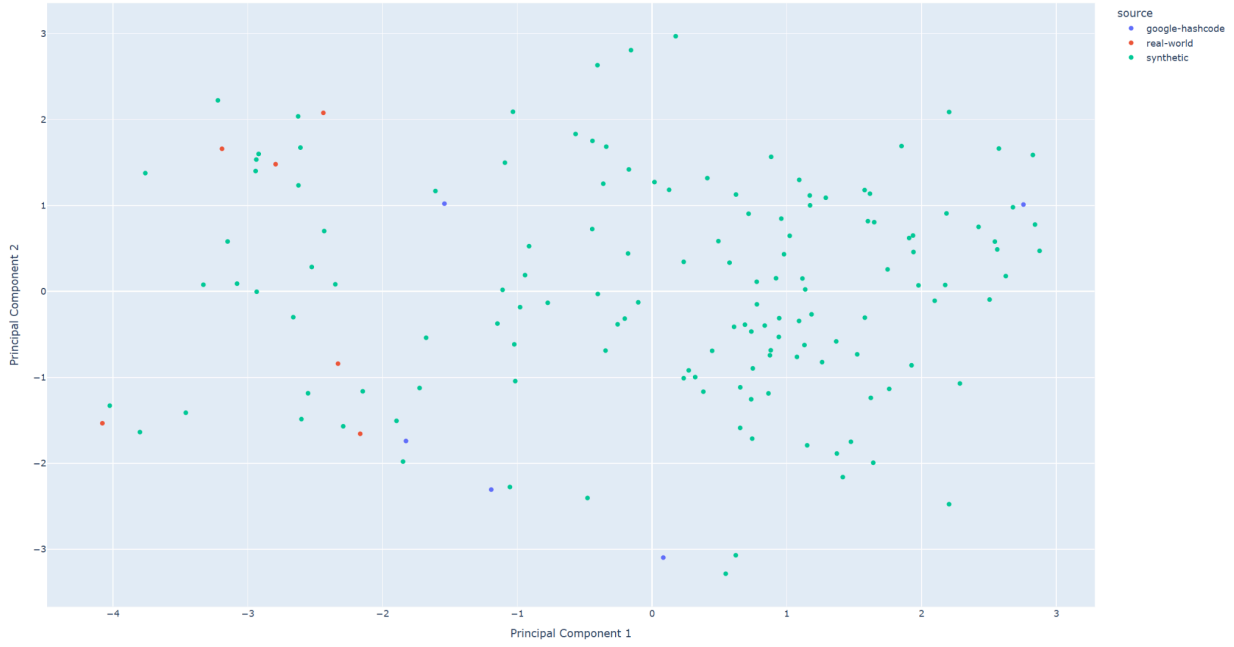

Figure 1. Instance features distribution

[12] https://github.com/uran-lajci/Book.Scanning.Dataset