Macau University of Science and Technology

School of Innovative Engineering

Software Engineering

SE360 Artificial Intelligence - Spring 2023

Group Project Report

Optimal Sample Selection

Abstract

A strong abstract sums up our work in very few sentences:

(i) The Algorithms about Optimal Sample Selection which can get optimal near optima l solutions

(ii) Friendly GUI to support our Algorithms

(iii) Database to record the results

(iv) Friendly User Manual which can guide our users to use our system

Keywords

Here are our keywords: **Artificial Intelligence; Greedy Algorithms; GUI ; Database; Time Trade-off;** eg.

Contents

1 Introduction

It is known that the amount of data generated has been increasing tremendously in the last few years due to the ease of accessing to the Internet, cheap or inexpensive mass storage devices, the ease of transferring data through Internet, communication lines and digital data are used in all walks of life. Nowadays, these big data have been used for data mining, knowledge discovery, machine learning, statistical learning, statistical analysis and experiments. In order to extract or discover useful data, information or knowledge from these big data, one of methods we usually adopted is samples selection.

This project we mainly focus on how to use less long string (k length) to convert short string (s length). And you can see detailed rules by this link.

We upload all of our source code on the github, and you can download it if you interested.[1]

2 Algorithms

2.1 Idea of Greedy Algorithms

Our algorithms mainly based on the ideas of Greedy Algorithms, it mainly statistic the number of short combination (s length) which long combination (k length) can covered. Then picking out the most *important* long string (k length). Then, the short combination and the long combination would be updated.

As Figure 2 we showed below(we provide a big size figure in the last of report, you can read it if you feel Figure 2 is not clearly), when n=5, k=4, s=3, our algorithms will statistic the number of the short combination and choose the long combination which has the most *"important"*, we find (1,2,3,4) has value 4 which is the most value. So we put it in Validation set and follow these steps over and over again until short combination set is empty.

The ideal is full, the reality is very skinny. If we want to use this idea, we need to overcome blew three problems:

[1] Our System's Repositories: https://github.com/IvanMao714/Optimal-sample-selection/

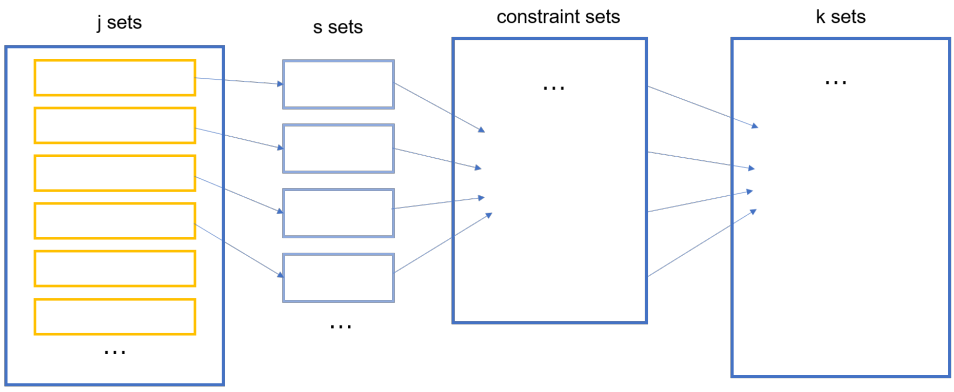

Figure 1: Basic ideas about

1. **How to trade-off time?**: When n and s are large, the algorithms will spend lots of time update value table.

2. **How to decrease the time of iteration?**: if we can't decrease the time about updating value table any more, whether can we decrease the times of iteration.

3. **How to deal with $j \neq s$ situation?**: When $j \neq s$, we find that how to pick out the short combination is a very important factor which will impact the number of long combination we want to pick out.

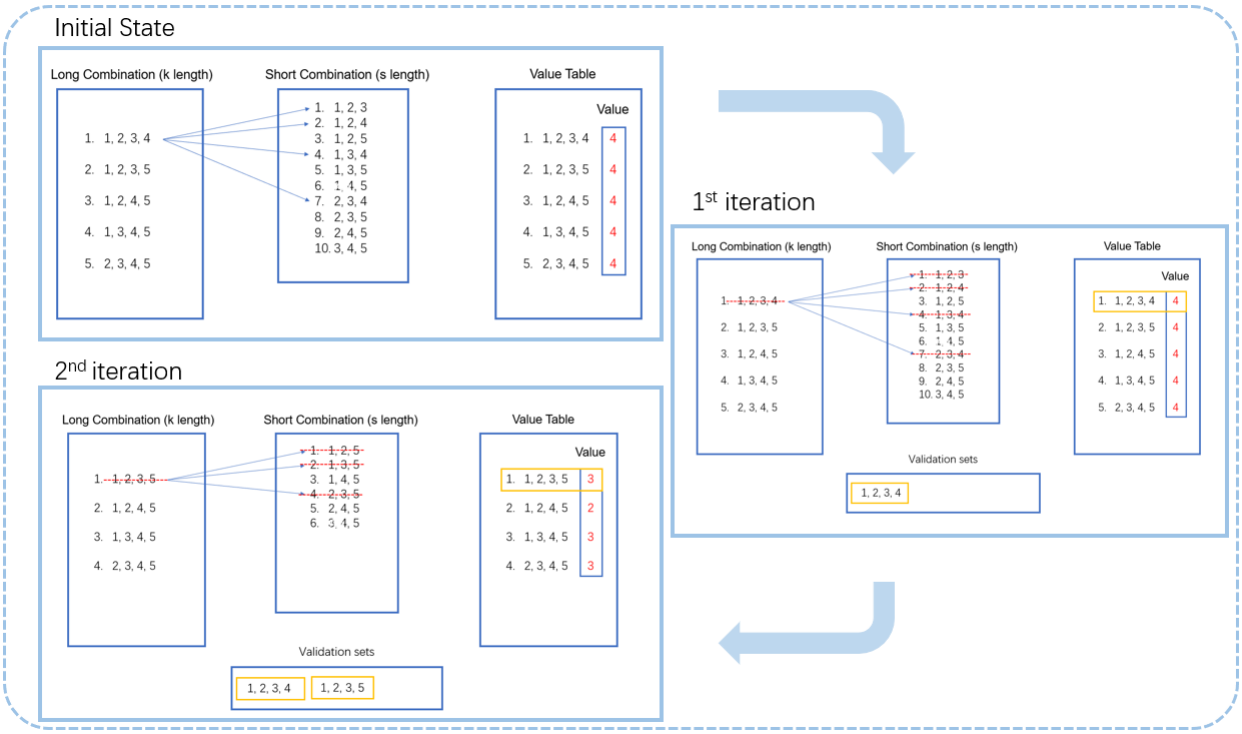

Figure 2: Basic ideas about our algorithms

2.2 Decrease the time of Updating the Value table

We find that the time of updating value table has huge change caused by n (length of long combination) and s (length of short combination). From Figure 3. And we find that s had more importance.Therefore, we can decrease the cost of time if we can number of items in value table.

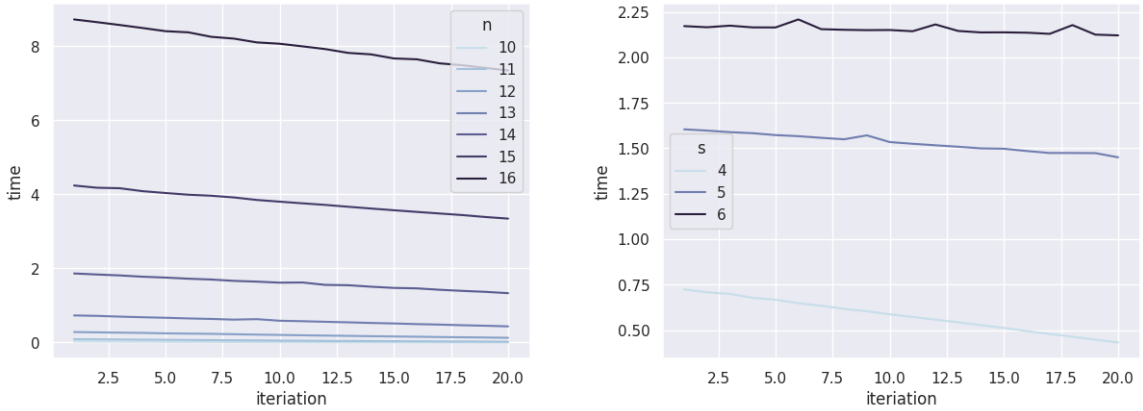

Figure 3: The Influence about time of updating value table

And it caused *cold start* which means values are most same in value table. It let algorithms to spend lots of time to compute useless data just like Figure 4.

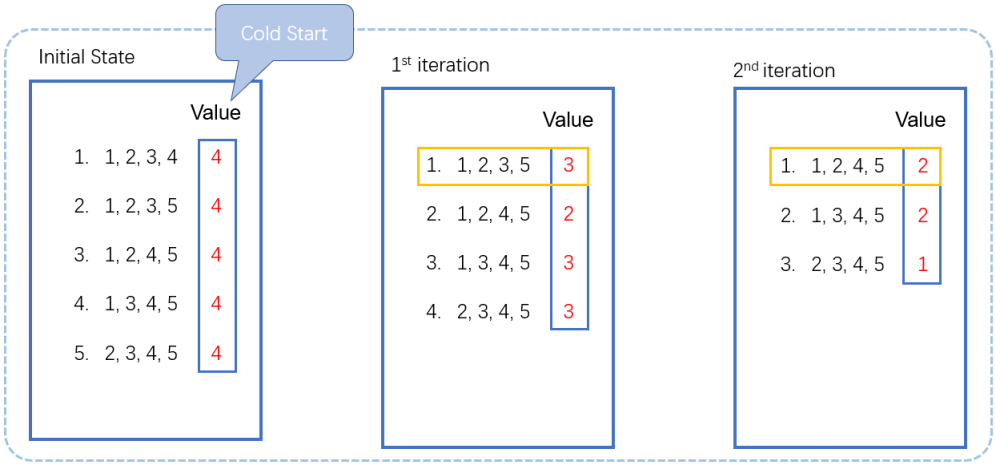

Figure 4: The situation about cold start

To solve this problems, we don't update value table by short combination set anymore. First, we divide short combination set into many segments. During segmenting, we ensure the range of each segment is large enough and smaller than total number. And we do it by segment short combination like Figure 5. It not only solve the problem about cold start, but also it actually decrease the time.

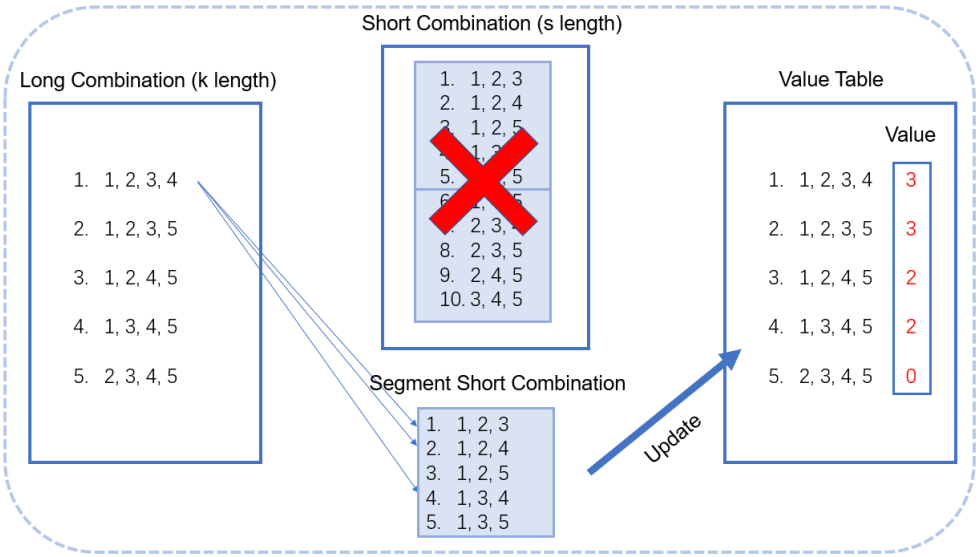

Figure 5: The Idea about using segment short table to update value table

Therefore, we divide short combination set into 3 segment until it number ≤ 400 After using this idea, our algorithms become faster than before and just like Figure 16 shown.

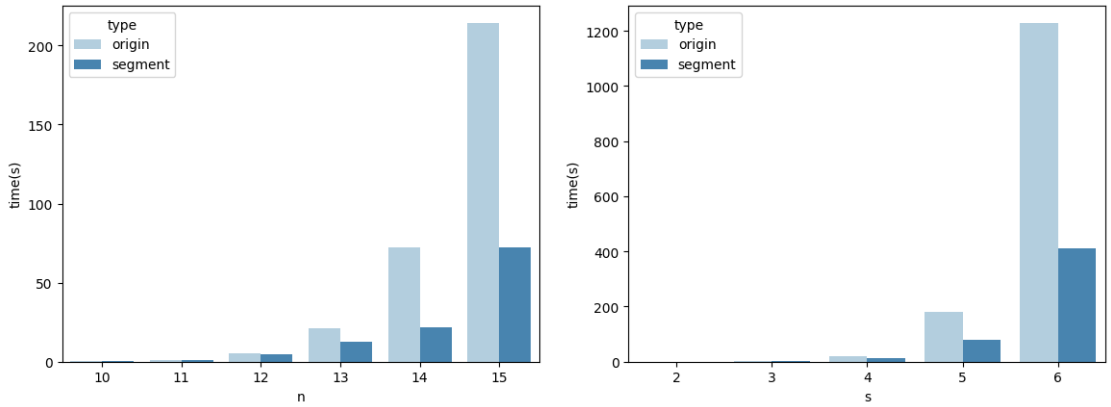

Figure 6: The effect of segment short combination

2.3 Multiply decrease Long Combination Set at once Update

We also think we accurate the algorithms if we can choose more Long Combinations as once. And the Figure 7 show our ideas.

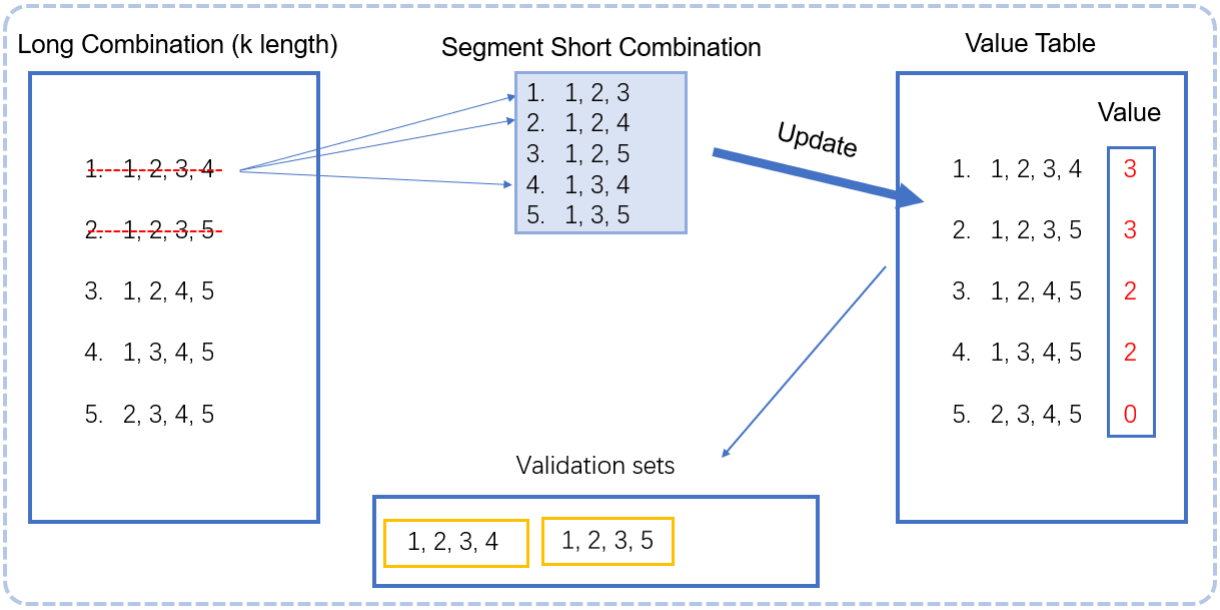

Figure 7: The idea of Multiply decrease Long Combination Set

And we find that the Long Combination Set has number of max value more than 1/10 number of Long Combination Set, it is the best to use this rules and trigger this mechanism. And we choose 1/125 number of number of max value(we called repeat times). If they do not cover then same short combination, it means they have the same and high ability. Therefore, we would choose these long combination with high performance. [8]

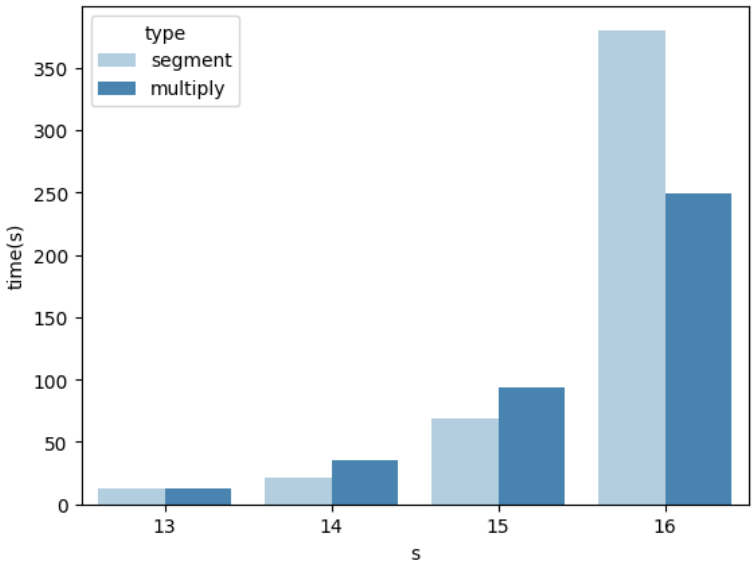

Figure 8: Effect of our idea

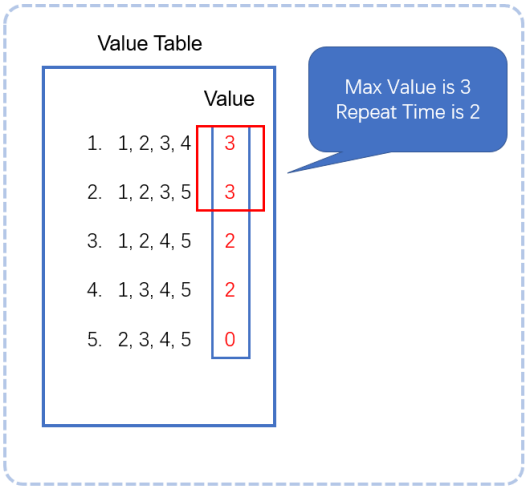

Figure 9: Definition of Multiply decrease Long Combination Set

2.4 Solution about $j \neq s$

Firstly, we choose the first short combination in J-sets possibly to produce a S-set, then all contents in S-set try to match J-sets reversely. The shorter combination which can cover the most number of J-set will be chosen. In Figure 10 ,as first 3Th short combination, we all choose (1,2) rather than choose (1,2)(1,4)(1,5). So we first statistic the number of occurrence about shorter combination in short combination, then we choose the shorter combination which has most times.

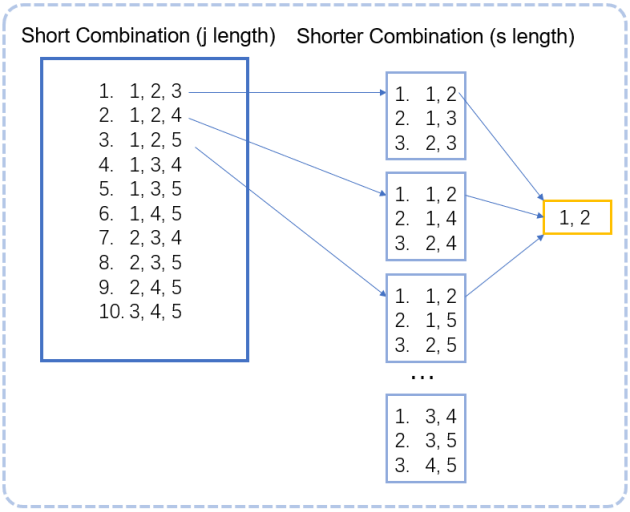

Figure 10: Idea about solution of $j \neq s$

Finally, it actually help us to get less optimal selection.

3 The result compare with example

We first compare our results with the guideline [13].

	Nearly optimal solution	Our Result	Time(s)
45, 10, 6, 5, 5	50	56	0.548
45, 12, 6, 5, 5	132	189	11.539
45, 13, 6, 5, 5	245	299	44.614
45, 10, 6, 4, 4	37	21	0.158
45, 12, 6, 4, 4	42	57	2.621
45, 16, 6, 4, 4	162	193	212.867
45, 11, 6, 6, 5	26	24	0.164
45, 12, 6, 6, 5	42	46	0.887
45, 16, 6, 6, 5	280	252	200.034
45, 12, 6, 5, 4	18	19	0.294
45, 13, 6, 5, 4	28	27	0.941
45, 16, 6, 5, 4	65	66	23.783
45, 12, 6, 6, 4	6	8	0.0749
45, 13, 6, 6, 4	12	10	0.191
45, 16, 6, 6, 4	38	24	3.147

Figure 11: Comparison our results with the guideline

The we compare our answer with test samples [13].

	Answer	Our Result	Time (s)
45, 7, 6, 5, 5	6	6	0.027
45, 8, 6, 4, 4	7	8	0.038
45, 9, 6, 4, 4	12	12	0.051
45, 8, 6, 6, 5	4	4	0.036
45, 10, 6, 6, 4	3	4	0.031
45, 12, 6, 6, 4	6	8	0.085

Figure 12: Comparison our results with the test samples

4 Shortcoming of our algorithms

Currently, the strategy of selecting J from J-set is not smart. In every times, we only choose the first J. It might cause a problem that we are likely to ignore J in other position, which means other J could produce a higher performance S.

Although we try our best to decrease the time and the time complexity is $O(C_n^k \times C_n^s)$, it is still large when n and s become bigger and bigger.Also, the greedy algorithm idea will cause

problems about local minimum, it is also influence our result.

5 GUI Design

In order to better experience our systems, we design friendly GUI which can support 3 edition systems(Windows, Mac OS, Android). And it can support random and specified selection.

We mentioned in our User Guide, so we don't explain our Gui function anymore.

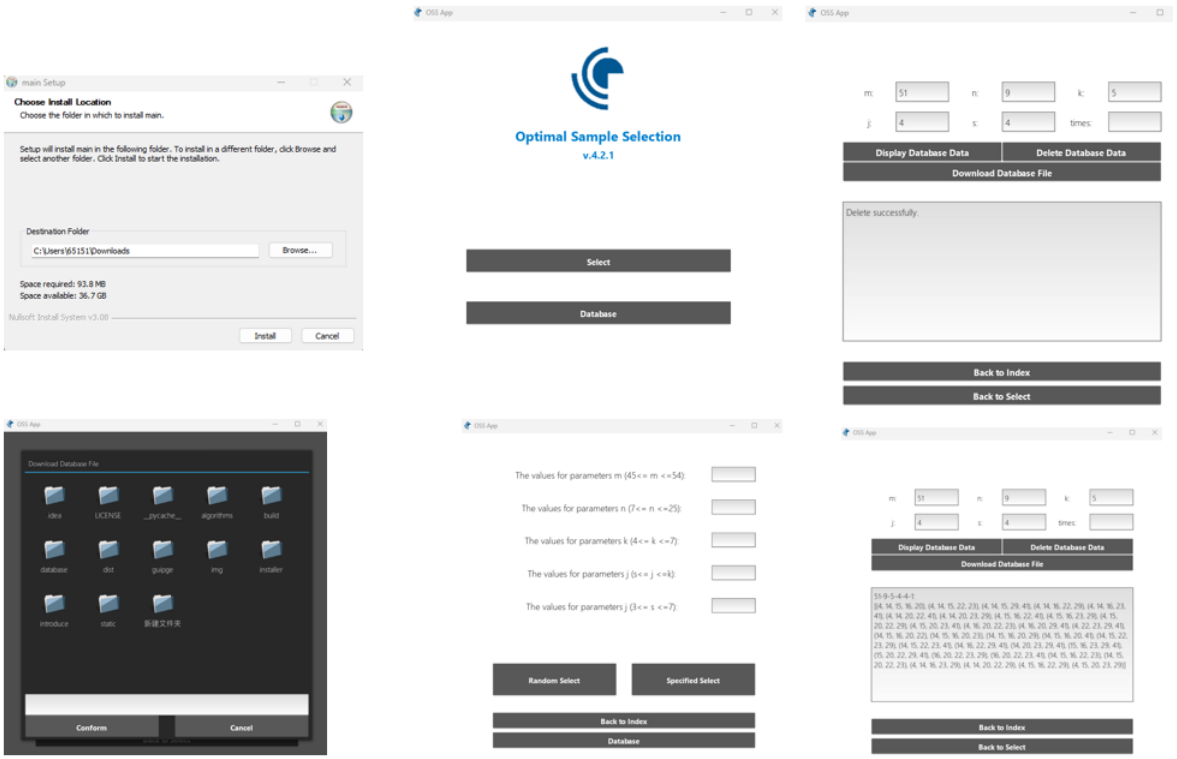

Figure 13: GUI Examples

6 Database Support

Our system also can record data had been selected on the server database. So user can easily query what their data, and they can download data in their computer.

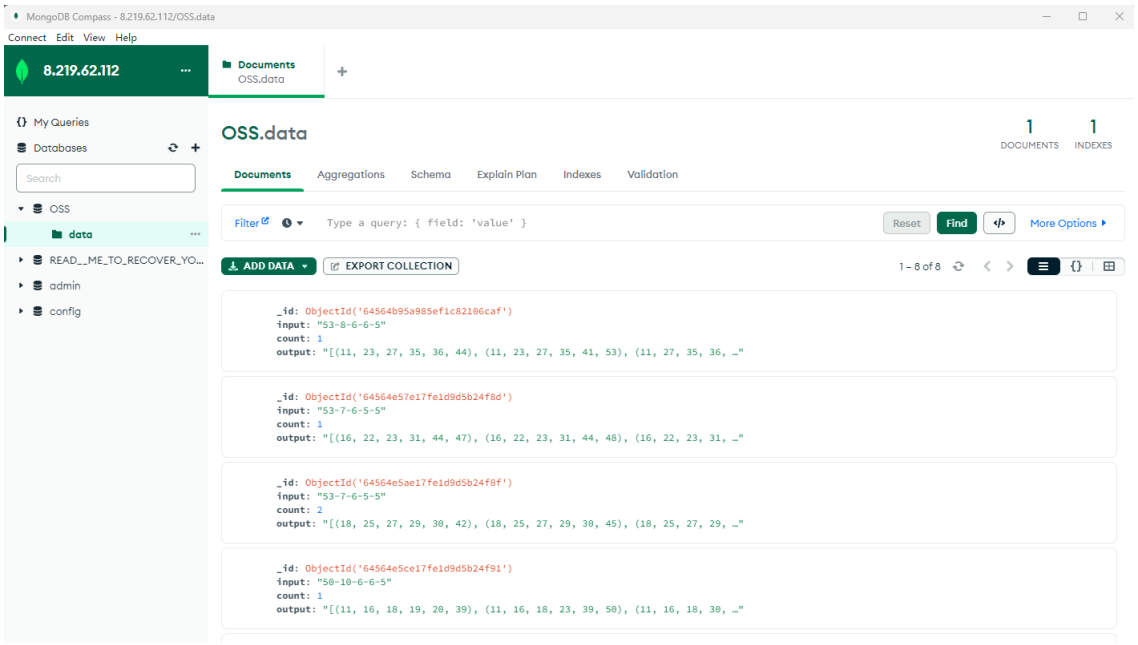

Figure 14: Our Database Server

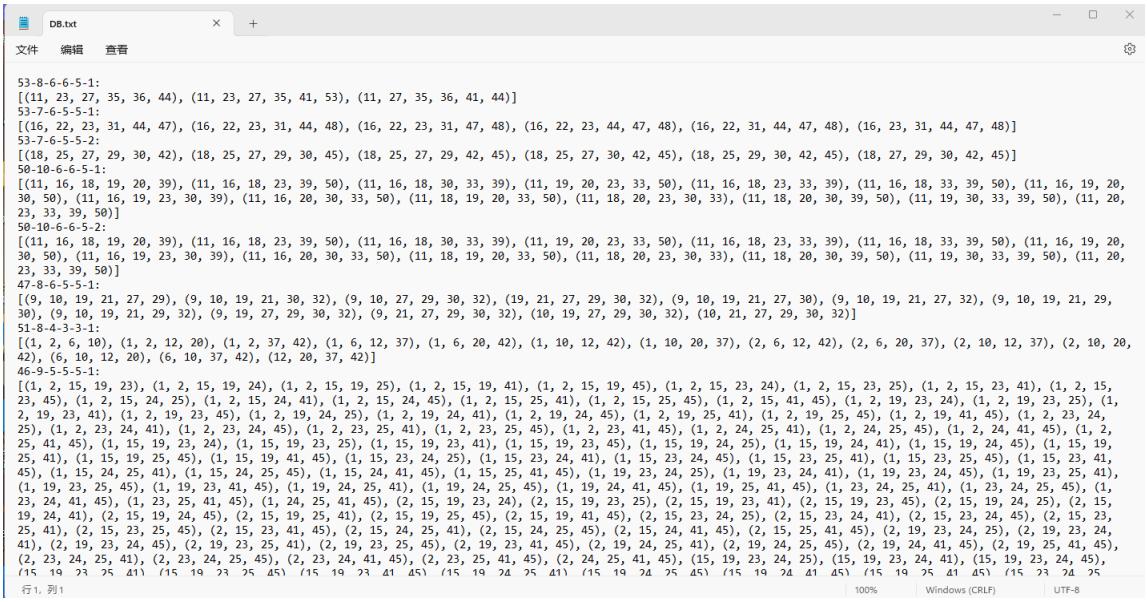

Figure 15: Local Database File downloaded by users

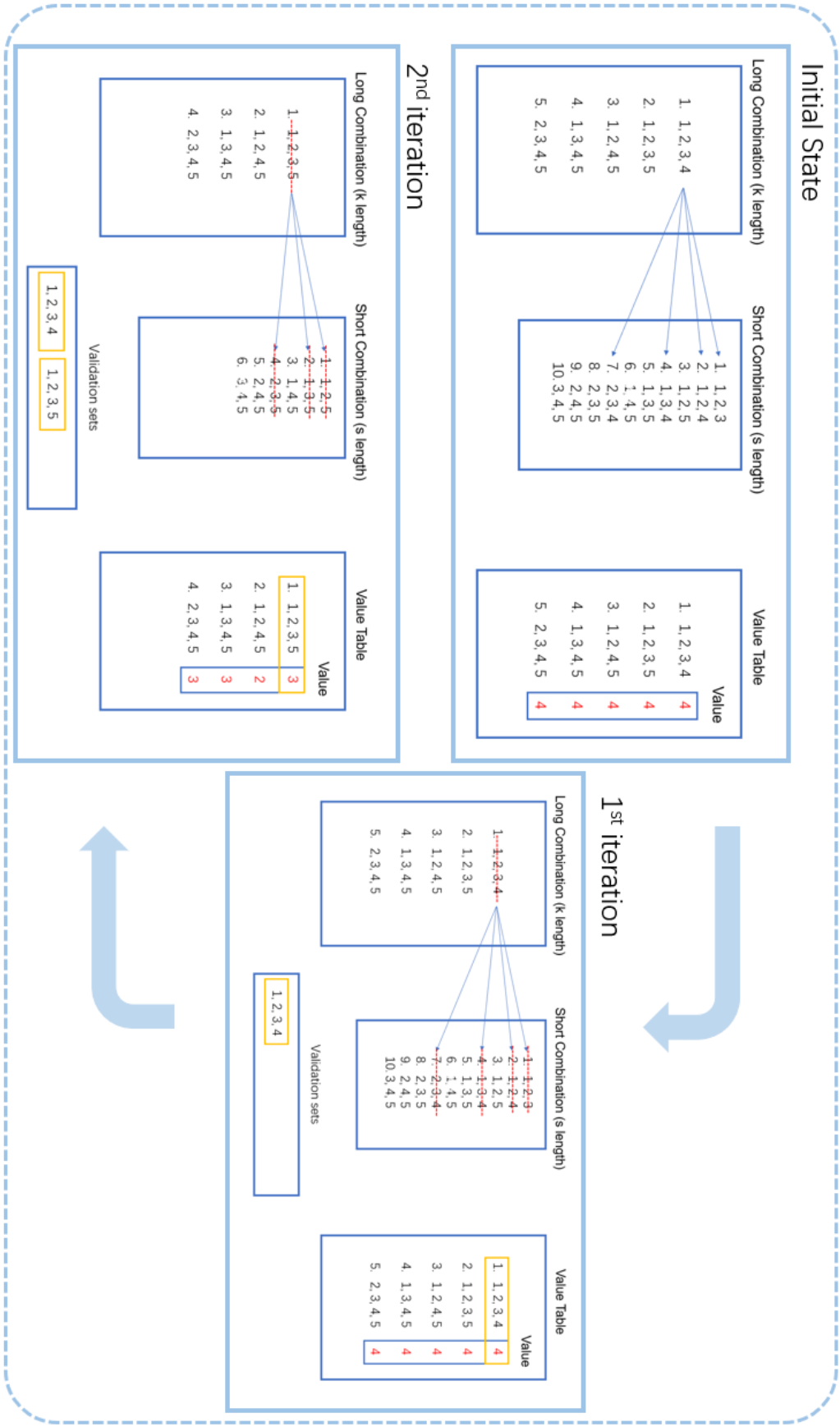

Figure 16: Basic ideas about our algorithms–Max Size