Cardinality Estimation

Student Number: 871089
Student Name: Zhuohan Xie

1. Introduction

AMS is one of the most famous algorithms for cardinality estimation, however, it normally overestimates the result and therefore needs to combine results generated from multiple copies of estimators. Median and mean are two common tricks to do the work, and they both have own merits and demerits, therefore, a combination of these two was proposed by the report, which takes mean of values around median (Movam) and is proven to outperform both tricks. Also, the report proposed an optimal AMS cardinality estimator, which is a good trade-off between accuracy, time and space and test it with different cardinality then find out AMS performs worse with higher cardinality.

2. Test Dataset

Testdata5 is chosen to explore trade-offs, which is generated by taking 500k distinct number from a large domain and copying each of them 1 to 5000 times randomly, then writing them to the test dataset in a random order. The other nine files are generated in a similar way and are used to test how cardinalities effect accuracy of AMS estimators.

Table 1: Test file details

File Name	Cardinality	Data Size
Testdata1	100000	230.2MB
Testdata2	200000	250.4MB
Testdata3	300000	246.1MB
Testdata4	400000	310.2MB
Testdata5	500000	343.3MB
Testdata6	600000	356.2MB
Testdata7	700000	383.9MB
Testdata8	800000	402.7MB
Testdata9	900000	457.7MB
Testdata10	1000000	487.5MB

3. Measuring Methods

- **Accuracy:** It is measured by median relative error, which is (estimated value – true cardinality)/ true cardinality and standard deviation. They show how much the results are underestimated/ overestimated and how stable the algorithms are separately.
- **Running Time:** It is the whole time taken by the program to output an estimation minus time taken by reading files.
- **Space:** It is measured by summing up all object size each algorithm created, since no space is released in algorithms of this report. And original algorithm requires $O(logn)$ bits for storing hash functions and $O(loglogn)$ bits to store estimated value z, therefore, it will need $O(k(logn + loglogn))$ for k estimators.

4. Accuracy Improvement Techniques

Considering algorithm for AMS, all measured values should start to form a Gaussian distribution with more estimators. As shown in Figure1, 100 and 200 estimators have been applied to a large file containing 1000 distinct numbers. Noticing that both two give median 10, and mean 10.3, 10.5 separately. By calculating $2^{z+\frac{1}{2}} = 1000$, it's easy to get the true value $z = \log_2 1000 - \frac{1}{2} \approx 9.47$. Median is good enough to filter out all extreme values and give us a fair estimated value that cannot be too far and mean can somehow give us a good chance to get very close to the true value, but not stable. Movam is a good approach to combine these two tricks, which is only considering values within 1 distance from median are important and taking mean of them for final value. In which case, 100 estimators give 9.93 and 200 estimators yield 9.77, which are both better than median or mean. Also, it shows the potential that more estimators are supposed to yield a more accurate value.

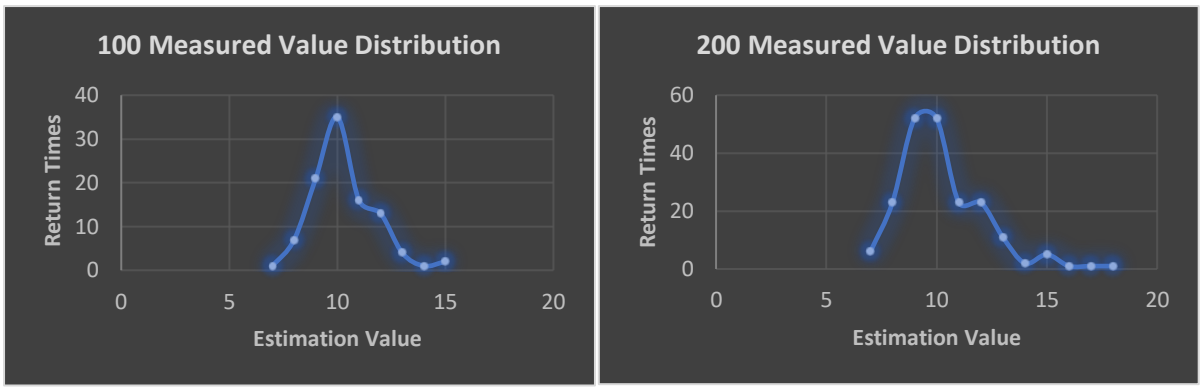

Figure1: Measured Value Distributions on 1000 distinct numbers

5. Experiment Results

In term of running time and space, the differences between three result combining techniques are slim, therefore, they are considered to consume same time and space as long as they use same amount of estimators.

5.1 Running Time/Space versus Estimator Number

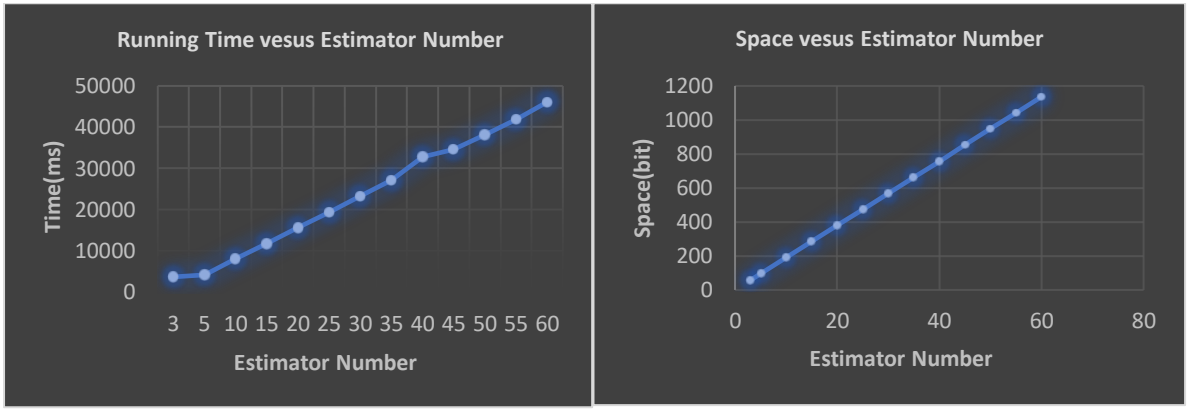

Figure2: Running Time/Space versus Estimator Number

As shown in the Figure above, running time and space grow linearly with estimator number, because with k estimators, the algorithm basically needs to perform k times of the original algorithm and needs k times of space to store all hash functions and estimated values.

5.2 Accuracy versus Running Time/Space

First of all, it's fair to say that Movam outperforms median and mean in both median relative error and standard deviation. Besides, standard deviations of all three techniques drop as running time/space grow, which means we can sacrifice time/space for stability. Also, it's worthy to point out that median always gives same median relative error and mean gets much higher relative error with limited time/space but is expected to get better and close to combination as time/space grows. Noticing that all median relative errors are positive numbers, and it can show that AMS algorithm tends to overestimate.

One good result achieved by experiments is applying ten estimators and using the combination approach to combine results, it yields close to 0.3 median relative error and only takes 8114ms and 189bits, however, the standard deviation is quite high, which means it cannot guarantee a good result all the time. Therefore, the optimal AMS estimator chosen from the report is to take 30 estimators with Movam combining trick. According to our experiments, this approach can guarantee a median relative error around 36%.

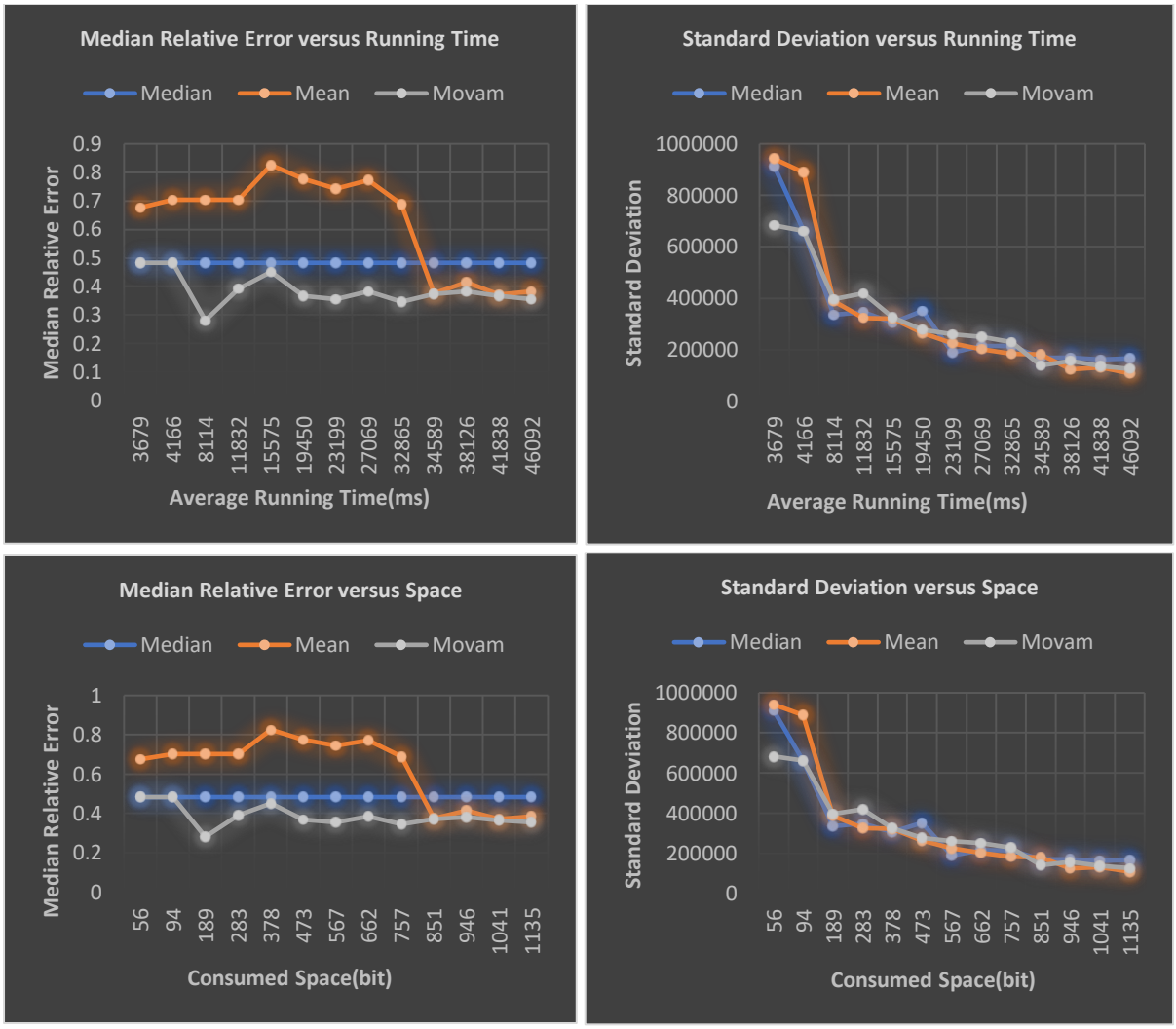

Figure3: Accuracy versus Running Time/Space

5.3 Accuracy versus Cardinality

The optimal algorithm is applied to all the datasets to explore how it performs with different cardinalities. It's clear that it works worse with higher cardinalities with respect to both median relative error and cardinality. Thinking of the return value formula for this algorithm, even if the differences between estimated values are slim but taking them to the pow of 2 will change drastically as cardinality grows.

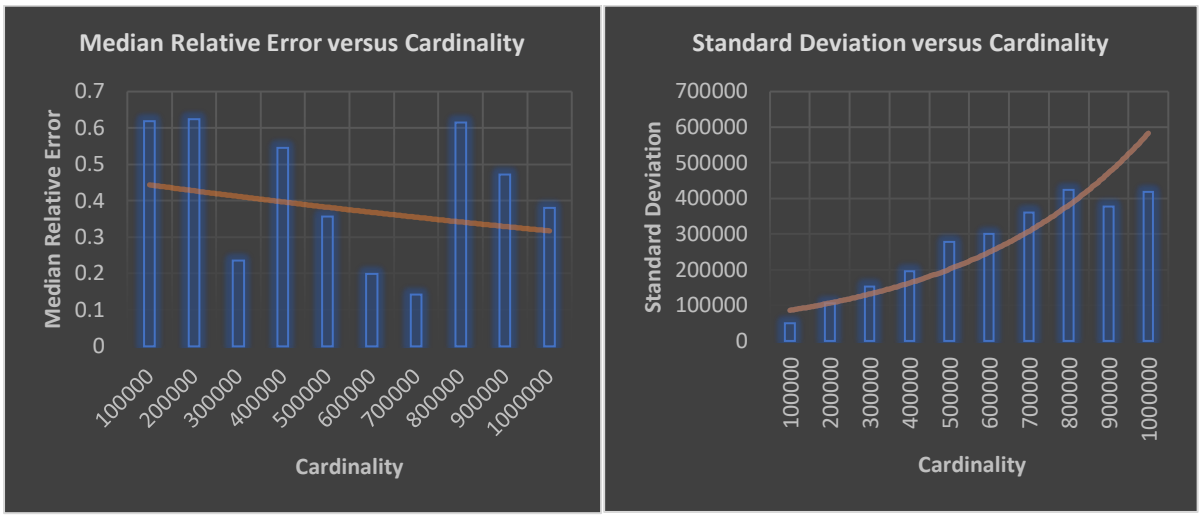

Figure4: Accuracy versus Cardinality

6. Conclusion

The report analysed general properties of AMS algorithms and explored trade-offs between accuracy, running and time, then proposed an optimal AMS based on abundant experiments, which is taking 30 estimators with Movam combining trick. And our optimal estimator is applied to different cardinalities and figured out this kind of algorithm doesn't work well with higher cardinality. Two potential future improvements would be signing more weights to values lower than median to eliminate bias toward large numbers and taking input data size and number into consideration and check how they can affect performs of AMS algorithms.