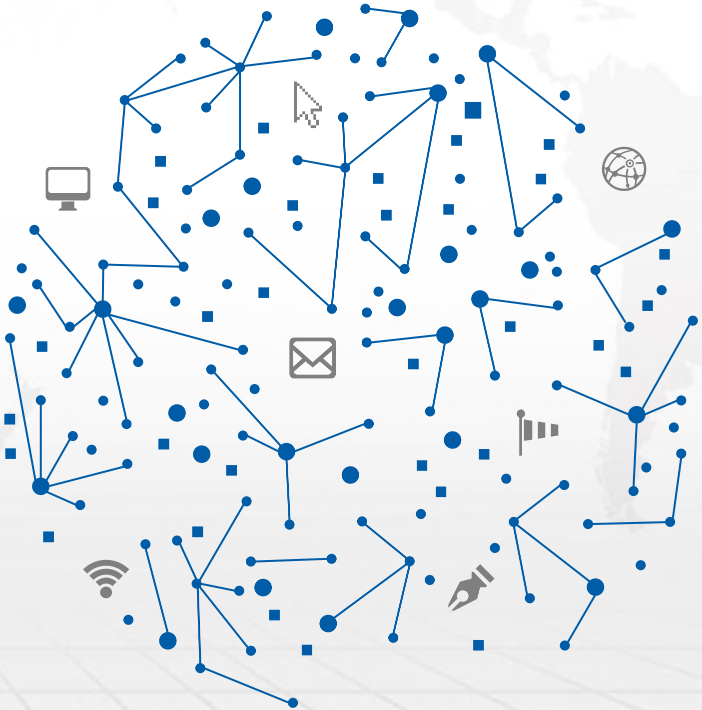
Challenges of managing LiDAR data, point cloud data

Group Member:
Yeqing Wang 862186226
Shiyi Zhang X674358



What are “Challenges” of managing LiDAR data?

What are “Challenges” of managing LiDAR data?

1. Acquisition

   How to acquire a huge amount of data, useful data?

2. Storage

   How to store them so that we can query them within a short time?

3. Utilization

   How to analyze and visualize the data?

Paper

"Dictionary compression in point cloud data management."

Pavlovic, Mirjana, et al.

Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2017.

Overview

1. Research Backgrounds
2. Proposed Methods
3. Experimental Evaluation
4. Conclusion

Research Backgrounds

Dictionary-Based Compression

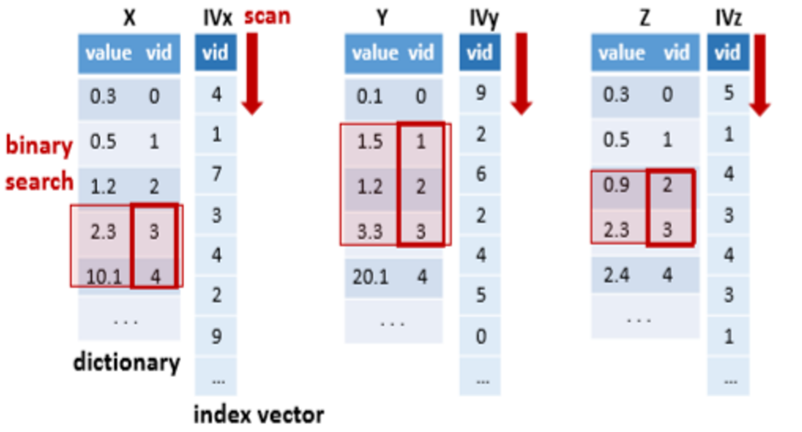


Figure 1: An example of range query execution over a dictionary-based representation of point cloud data.

Advantages: encode the data using information that is not stored in the actual datafile we are compressing

How to optimize?

challenges: waste computational resources (it does not leverage the spatial properties of data)

Dictionary-Based Compression

Dictionary Compression in Point Cloud Data Management

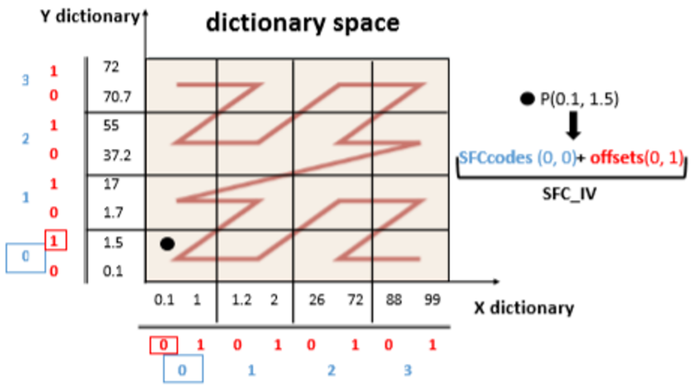
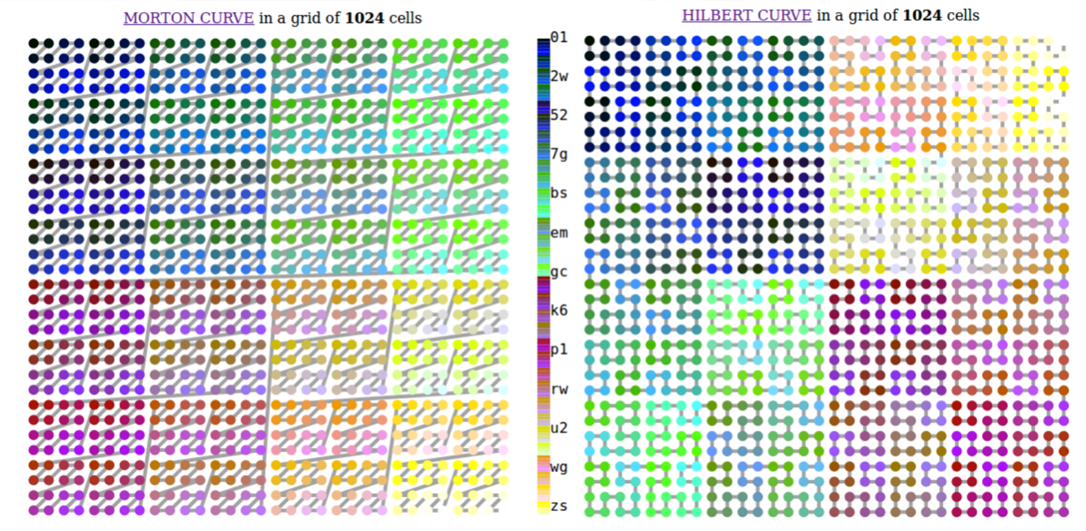


Figure 2: Dictionary space, 2D illustration.

How to optimize: restrict a search range using an index structure (that combines all 3 dimensions) and improve the data access patterns

Challenge: it is hard to achieve this in both time and space-efficient manner

Space-filling Curves



Advantages: transforms data from a multi- dimensional to a one-dimensional domain using SFC to impose a total, 1 D order by visiting all the points in a d-dimensional grid exactly once.

Space-filling Curves

SFC order reorganizes data (three steps)

Step1

1

Partition the dataset’ s universe with a uniform grid and assign to each cell a value on the space-filling curve

Step2

2

Assign SFC code to every point cloud entry according to the grid cell they belong to, where multiple point cloud entries can map to the same SFC code value curve

Step3

3

Sort the points based on the assigned SFC code

Space-filling Curves

Range query execution order (two steps)

Step 1

Transform a query to the 1D domain according to the SFC-order and perform binary search on the SFC codes data structure based on the transformed ranges

1

2

Step 2

As a SFC code is assigned per cell and not per point basis, all the points whose SFC code matches the result of the binary search have to be additionally checked whether they belong to the query range in order to remove false positives.

Space-filling Curves

Challenge:

the SFC codes structure requires additional storage resources. (hurts space efficiency)

Space-filling Curves
Dictionary-based Compression

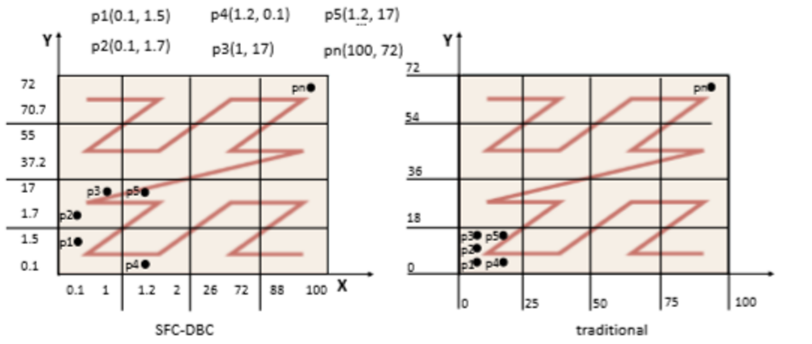


Figure 3: SFC-DBC (data-oriented) and SFC-based (space-oriented) partitioning strategy.

Advantages: SFC-DBC combines DBC with SFC order to ensure space efficiency and preserve spatial proximity, thus optimizing for query execution

Proposed Method

Space-filling Curves
Dictionary-based Compression

3. Additionally store the position of the point within the cells

1. producing a 3D dictionary space and a grid on top of it

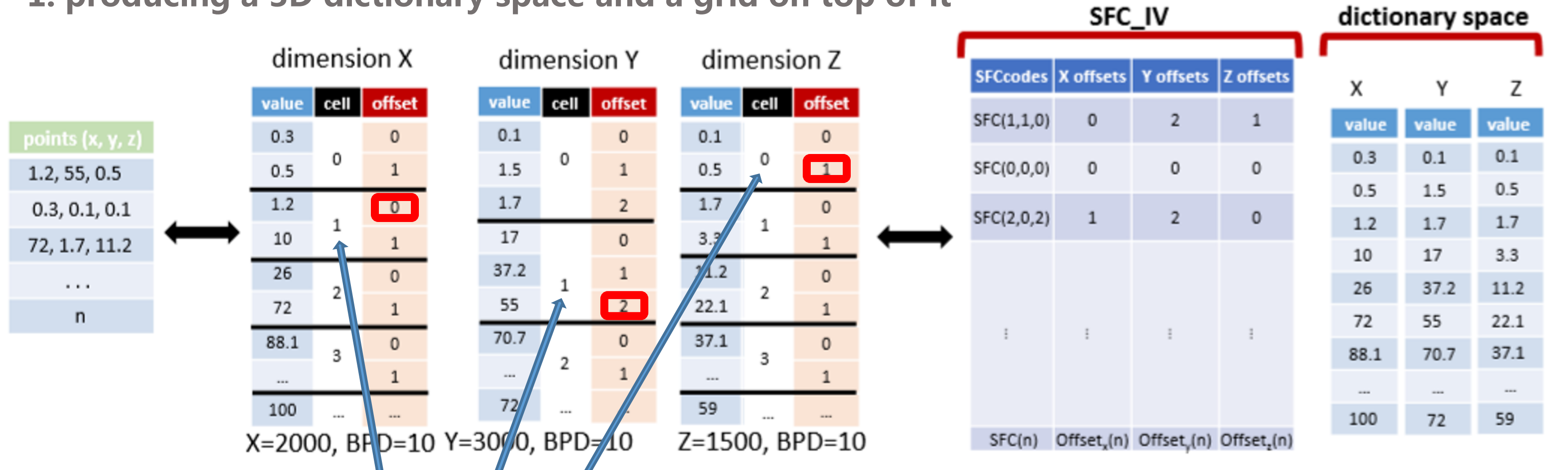


Figure 4: Point cloud data organized according to SFC-DBC Encoding.

2. Assign a SFC code to every point according to the dictionary cells they belong to

4. Once the final structures are produced, we sort them according to the assigned SFC code

Query Execution

First step:
Produce a candidate result set.

Algorithm 1: Query Execution: produce candidate results set

Input: q: range query - defined with two coordinates
Output: minDQ, maxDQ: min and max position in dictionary that corresponds to query range
Output: candidateSet: candidate result set

//transforms query to 1D:
for $d = 0$ **to** *dimensions* **do**
  $minDQ[d] = binaryS(dictionary[d], q.low[d])$
  $maxDQ[d] = binaryS(dictionary[d], q.high[d])$
end
$qSFCcodes = calcSFCcode(q, minDQ, maxDQ)$
$candidateSet = binaryS(qSFCcodes, SFCcodes)$
return *candidateSet*

Query Execution

Second step:
Produce the final result set.

Time consuming:
scan offsets + decoding SFCcode

Algorithm 2: Query Execution: produce final results set

```
Input: q: range query - defined with two coordinates
Input: candidateSet: candidate result set
Input: minDQ, maxDQ: min and max position in dictionary
       that corresponds to query range
Input: EPC: number of entries per cell, d - dimension
Output: pOut: point cloud result set

for i = value in candidateSet do
    cell_id = decode(SFCCodes[i], d)
    base = cell_id * EPC[d]
    //retrieve the positions of the points for the given SFCcode
    <inputMin, inputMax> = mapSFCcodeToInputPosition(i)
    //enclosedByQuery condition
    if minDQ[d] < base AND (base + EPC[d]) < maxDQ[d]
     then
        pOut.setRange(inputMin, inputMax)
        continue
    end
    //not enclosedByQuery - retrieve offsets
    for j = inputMin; j < inputMax do
        position = base + offests[j];
        if minDQ < position < maxDQ then
            pOut.set(j)
        end
    end
end
return pOut
```

Analyze

Space requirements
Assume each dictionary in 3d space has the same length

Traditional DBC: $3 \times (DS \times de + n \times log_2 DS)$

SFC-DBC: $3 \times (DS \times de + n \times log_2 \lceil DS/2^{BPD} \rceil + \#SFCcode \times BPD)$

DS: the number of entries
de: the size of an entry
n: the number of points
$log_2 DS$: the number of bits per Index Vector entry

BPD : the number of bits assigned per dimension
#SFC-code : represents the number of distinct SFCcode values

Experimental Evaluation

Experiment Environment

Hardware Configuration

SuSE Linux Enterprise Server 12 SP1 machine

4 Intel Xeon CPU E7-4880v2 processors at 2.50GHz and 512GB of RAM

Each processor has 15 cores with private L1 (32KB) and L2 (256KB) caches, as well as 37,5MB of shared L3 cache.

1

SAP HANA

HANA is an in-memory database that offers the possibility to store data in either a row-oriented or a column-oriented fashion.

2

Experiment Environment

Dataset

1. AHN2 dataset
2. Senatsverwaltung für Wirtschaft, Technologie und Forschung" and "Europäischer Fonds für regionale Entwicklung (EFRE)" provided the dataset that are generated by using dense image matching.

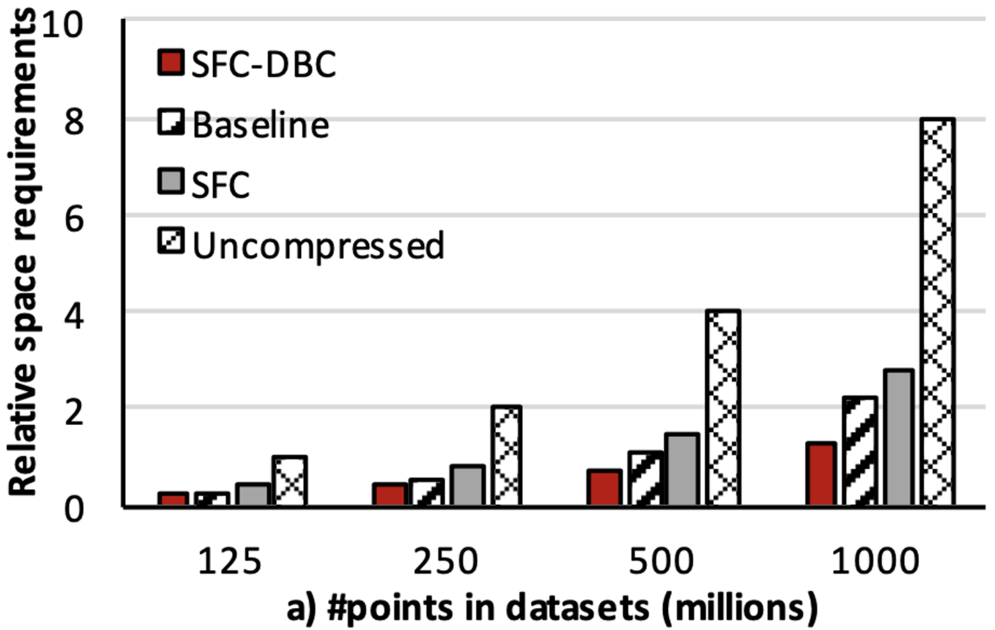
1

Query

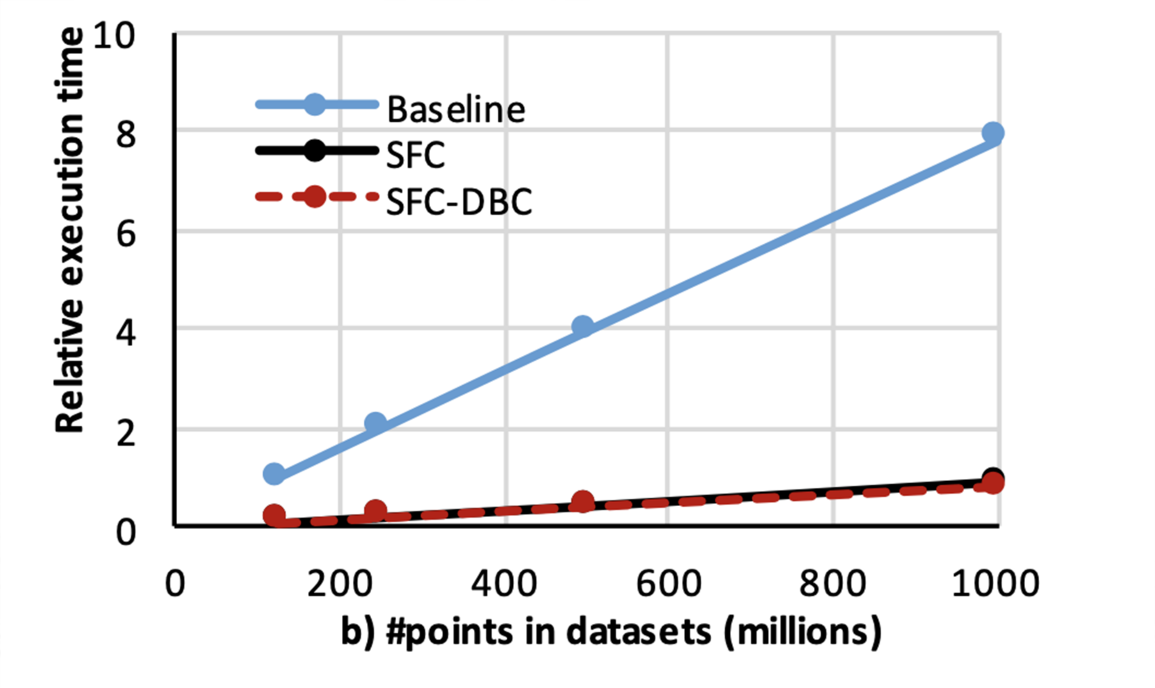
one hundred 2D and 3D range queries that follow uniform distribution.

2

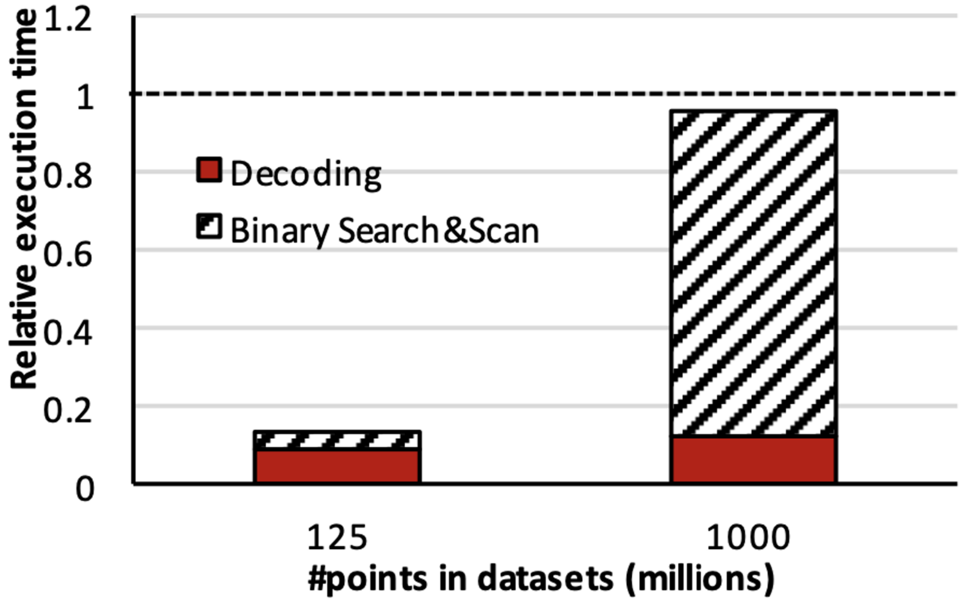
Space Requiremets



Query Performance



Query Performance



Impact of Skew

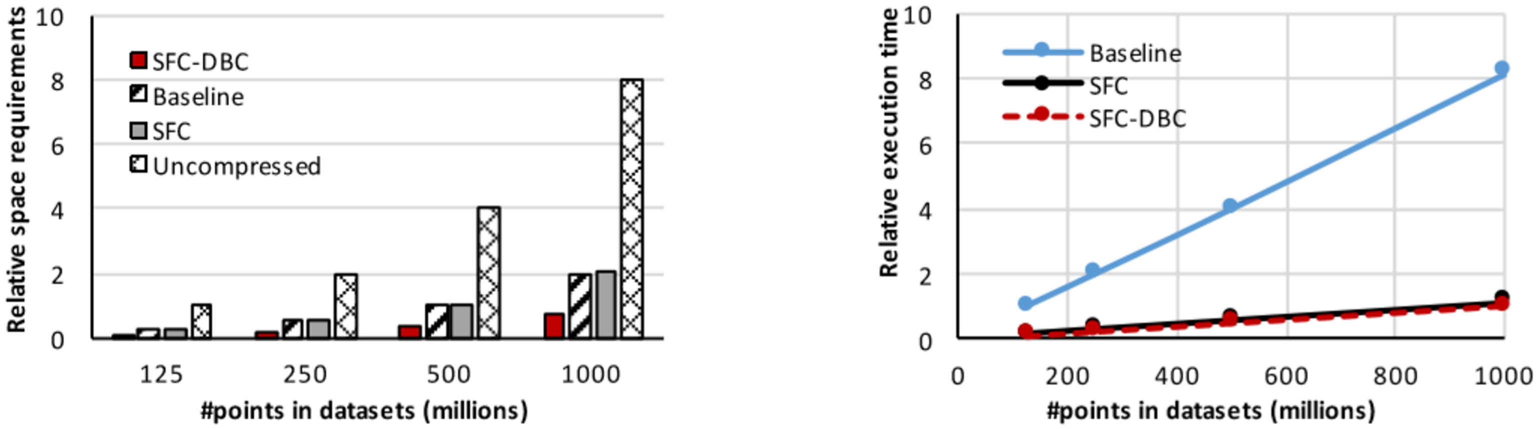


Figure 9: The impact of skew: space requirements and query execution time.

Impact of Filtering

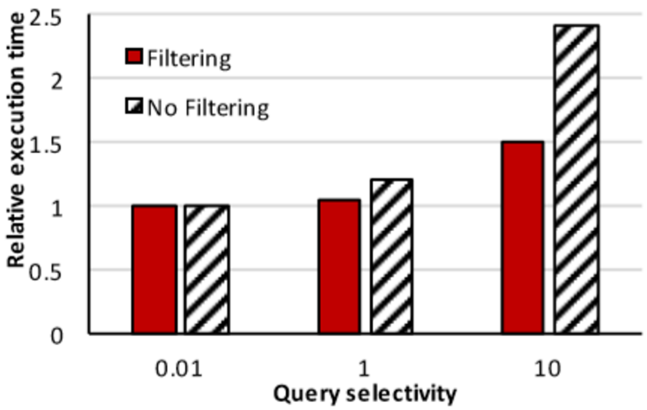


Figure 11: SFC-DBC: impact of filtering.

The relative execution time for the SFC-DBC approach, when enable and disable filtering

The filtering step significantly improves the execution time for low selectivity queries, considering that it filters more data (e.g., the improvement in the execution time is 38% for 10% selectivity)

Filtering does not have a significant impact on performance when executing high selectivity queries (e.g., for 0.01% selectivity queries the improvement in the execution time is 0.6%).

Conclusions

Conclusion

Executing existing data management solutions face two challenges: time and space efficiency.

SFC-DBC employs dictionary-based compression in the spatial data management domain, enhancing it with indexing capabilities without introducing additional storage overhead, thus solve the two problems.

Thanks for listening!