# A Study on Visualization from Large-Scale Datasets

Yiwei Liu

yiweiliu@yorku.ca

## Abstract

Visualizing the whole large dataset of two dimensions is often infeasible due to limited rendering power or storage constraints in most visualization tools. Hence, a visualization-representative subset is usually selected from of the original dataset. In order to find such subsets, this project follows the research [10] to formulate the visualization quality decay as an optimization problem. Then, this project studies six methodologies to solve this problem. This project starts from implementation and evaluation on these methods on small-scale datasets. After a preliminary assessment on these methods, we select three promising ones and compared their performance on visualizing large-scale datasets. Final experimental results show that Interchange Search Heuristic outperforms other approaches in finding subsets with good visualization quality if it is given enough time.

## 1 Introduction

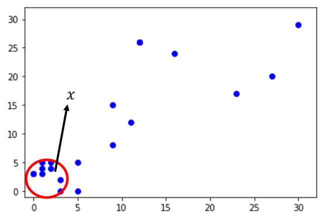
Data visualization can provide much information in a straightforward way and helps further data analysis. However, when coming to big data visualization, it usually takes much time for most existing visualization tools to produce satisfying results for users. Besides, some of these tools cannot handle with too much data because of their limited storage space. Hence, how to speed up big data visualization without decreasing the quality is a popular research topic in recent years. The idea behind many relevant studies is data reduction, which means we need to choose part of the original dataset. Incremental visualization [5] and binned aggregation [7] are two representative methods of data reduction. Incremental visualization keeps refining visualization results from the previous one step so that the visualization tool only takes a small proportion of data in every step. However, this method requires much sampling budget. Binned aggregation divides data into bins, and aggregates some data from each bin according to certain rules. But this approach is lack of feasibility because it depends on predefined bins to partition data. A recent study [10] also focuses on data reduction for visualization from a perspective of mathematical optimization. Given a fixed size of reduced dataset, this study adopts Interchange Search Heuristic to quickly find a subset with good visualization quality. To validate this approach, as well as evaluating alternative methods for the same optimization problem, this project implemented six algorithms and assessed their performance on large-scale and two-dimensional dataset visualization.

## 2 Methodologies

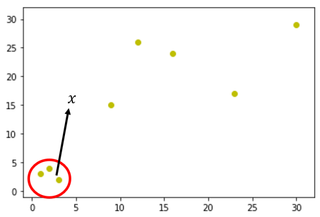
This section starts from formulating the visualization quality decay caused by data reduction as an optimization problem. After analysing its complexity, this problem is found to be NP-hard and equal to a famous topic in the research of graph theory [10]. Then, three methodologies about this research topic are briefly discussed.

## 2.1 Problem Formulation

It is obvious that we are to have visualization quality decay after data reduction. This decay can be seen as visual difference between the figure of points from the original dataset and its counterpart from the subset. To formulate this decay, we can start from studying a region around a point in a plot. Figure 1 shows two plots generated from the original dataset and its subset respectively. For simplicity, we name the original dataset as groundtruth in the following report. Considering a region around point $x$ in the subset plot, the more points from the groundtruth we have in this region, the more similar this region is to the plot of groundtruth.



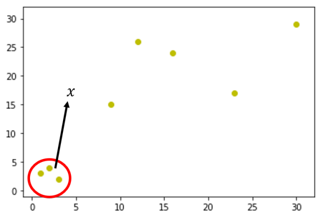
**(a) Groundtruth**



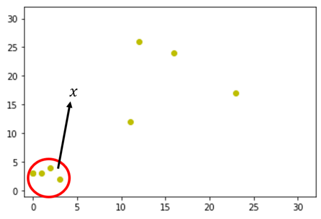
**(b) Subset**

**Figure 1: Plots of Groundtruth and Subset**

Then we can use total proximity of point $x$ in the plot and all the dataset points to measure the number of points within a region of $x$. Large proximity means that many points from the groundtruth fall in this region. To calculate this proximity, we use a function $k(x, y)$, where $x$ and $y$ stands for two



**(a)** $S_0$



**(b)** $S_1$

**Figure 2: Plots Generated by Different Strategies**

points. Figure 2 compares two plots produced by different data reduction strategies $S_1$ and $S_2$. There is no much visual difference between regions around $x$ in these two plots, though $S_2$ makes $x$ surrounded by more points from the groundtruth. Hence, we can conclude that too many points around $x$ does not help much to reduce the visualization quality decay. Now, a loss function for each point in a plot can be defined as

$$\textit{point loss} = \frac{1}{\sum_{\mathbf{s_i} \in S} k(\mathbf{x}, \mathbf{s_i})}$$

$S$ refers to a subset selected from the groundtruth and $\mathbf{s_i}$ is a point in this subset. Points in all equations of this project are vectors in $R^2$. $f(x) = \frac{1}{x}$ in this formulation corresponds to the conclusion from Figure 2. Next, the point loss can be rewritten by Second Taylor Expansion as follows.

$$\frac{1}{\sum_{\mathbf{s_i} \in S} k(\mathbf{x}, \mathbf{s_i})}$$
$$= 1 - (\sum k(\mathbf{x}, \mathbf{s_i}) - 1) + (\sum k(\mathbf{x}, \mathbf{s_i}) - 1)^2$$
$$= \sum k(\mathbf{x}, \mathbf{s_i}))^2 - 3 \sum k(\mathbf{x}, \mathbf{s_i}) + 3$$

So far we only obtained the visualization quality loss function of each point in a plot. However, for measuring the quality loss of the whole plot, we need to integrate the point loss function over all points in this plot, so we have

$$plot\ loss = \int \sum k(\mathbf{x}, \mathbf{s_i}))^2 - 3 \sum k(\mathbf{x}, \mathbf{s_i}) + 3\, d\mathbf{x}$$

There are two constants in the plot loss function, $\int \sum k(\mathbf{x}, \mathbf{s_i})\, d\mathbf{x}$ and $\int \sum k(\mathbf{x}, \mathbf{s_i})^2\, d\mathbf{x}$. Because integration goes through every point $x$ in the plot and total proximity of these points and subset points is not affected by the choice of subset. After eliminating these constants, we have a compact plot loss function

$$\int \sum_{\mathbf{s_i}, \mathbf{s_j} \in S; i<j} k(\mathbf{x}, \mathbf{s_i}) k(\mathbf{x}, \mathbf{s_j})\, d\mathbf{x}$$

If we reorder integration and summation in the equation above, we have

$$\sum_{\mathbf{s_i}, \mathbf{s_j} \in S; i<j} \int k(\mathbf{x}, \mathbf{s_i}) k(\mathbf{x}, \mathbf{s_j})\, d\mathbf{x}$$

The component $\int k(\mathbf{x}, \mathbf{s_i}) k(\mathbf{x}, \mathbf{s_j})\, d\mathbf{x}$ is equivalent to the proximity of two subset points $\mathbf{s_i}$ and $\mathbf{s_j}$. So we have the final loss function as

$$\sum_{\mathbf{s_i}, \mathbf{s_j} \in S; i<j} \tilde{k}(\mathbf{s_i}, \mathbf{s_j})$$

where $\tilde{k}$ is also a proximity function of two points.

Our objective in this project is to minimize this loss so that we will have visualization quality decay as little as possible after reducing data from the original dataset. Suppose the subset size is a given number $K$, that is, $|S| = K$, then the objective function is

$$min \sum_{\mathbf{s_i}, \mathbf{s_j} \in S} \tilde{k}(\mathbf{s_i}, \mathbf{s_j}) \qquad (1)$$

To ensure the convexity of this objective function, $\tilde{k}(\mathbf{s_i}, \mathbf{s_j})$ is defined by an exponential function $e^{\frac{-\|s_j - s_i\|^2}{2\epsilon^2}}$, where $\epsilon = \frac{max\|\mathbf{s_i} - \mathbf{s_j}\|}{100}$. The chose of $\epsilon$ here also follows [10].

## 2.2 Complexity Analysis

Complexity analysis of the objective function starts from mapping this function into a graph. First let us denote $w_{ij} = \tilde{k}_{max} - \tilde{k}(\mathbf{s_i}, \mathbf{s_j})$, then we can construct an undirected weighted graph $\mathcal{G}'$ with $K$ vertices and $w_{ij}$ the weight of the edge between vertices $i$ and $j$. $\mathcal{G}'$ can be also seen as part of a graph $\mathcal{G}$ with $D$ vertices, where $D$ corresponds to the number of points in the original dataset to be visualized.

Now we can say our optimization problem 1 is equal to $max \sum_{i,j \in \mathcal{G}'} w_{ij}$, which is the formulation of the Densest $k$-Subgraph (DkS), a famous problem in the field of graph theory research. This problem aims to find a subgraph which sum of weighted edges is maximal. But the DkS problem is NP-hard [1]. Methods to solve a NP-hard problem usually fall into three categories. The next subsection will talk about these categories in general.

## 2.3 Methodologies Descriptions

Approaches to handling with NP-hard problems can be categorised into algorithms for finding exact solutions, approximation algorithms, and search heuristic. Table 1 shows some recent methodologies for the DkS problem.

In terms of finding exact solutions, some researchers consider DkS as a submodular function to be minimized and proposed an algorithm with low computational complexity [6]; some other researchers modify the basic ILP approach to solve the DkS problem in large graphs [11]. Tabu search heuristic [8] is proved to be efficient in small graphs but

**Table 1: Related Work on DkS**

| Exact Solution | Approximation | Search Heuristic |
|---|---|---|
| Binary Linear Programming (BLP) [4] | $\frac{1}{2}$-approximation [1] | Interchange [10] |
| Submodular Function Minimization [6] | $\frac{1}{4}$-approximation [3] | Tabu [8] |
| Large-scale Optimization for DkS [11] | | |

experiments on applying this search heuristic on larger or denser graphs need to be carried on.

This project only focuses on implementing and evaluating Binary Linear Programming (BLP), $\frac{1}{2}$ - Approximation, and Interchange search heuristic due to time constraint. Each of these methods will be briefly discussed in the next subsections.

*2.3.1* ***Binary Linear Programming*** The standard linear programming form of the DkS problem is

$$max \sum_{i,j} w_{ij} b_{ij}$$

$$\sum a_i = K, \qquad b_{ij} \le a_i, \qquad b_{ij} \le a_j,$$

$$a_i \ge 0, \qquad b_{ij} \ge 0, \qquad a_i \in \{0,1\},$$

$$b_{ij} \in \{0,1\}$$

Since variables in this problem are binary, this problem will be solved by Binary Linear Programming (BLP) techniques. A common practice of dealing with BLP is using the dual simplex method and branch-and-cut [2]. Branch-and-cut is an algorithm combined with branch-and-bound and the cutting plane method, and this algorithm is used in *cvxopt*, a powerful python library for solving optimization problems [9].

*2.3.2* $\frac{1}{2}$ ***- Approximation*** Since it may take much time to find exact solutions to DkS, approximation algorithms are often used to find a hopefully near-optimal solution. If the value of the optimal solution to a maximization problem is denoted as $O(Opt)$ and the value of an approximated solution is $O(S)$, then the solution produced by Algorithm 1 satisfies $\frac{1}{2}O(Opt) < O(S)$. This algorithm has a complexity of $O(n^2 + K^2 \log K)$ if heaps are used in its implementation.

**Algorithm 1** $\frac{1}{2}$ - Approximation Algorithm

```
$S = \emptyset$, subgraph size $k$
while $|S| \le k - 2$ do
    Select $(v_i, v_j)$ whose weight is the maximum in the edge set of $\mathcal{G}$
    $S = S \cup \{v_i, v_j\}$, and delete $v_i, v_j$ from $\mathcal{G}$
end while
if $k$ is an odd number then
    Add an arbitrary vertex to $S$
end if
```

*2.3.3* ***Interchange Search Heuristic*** Another method to solve DkS is Interchange search heuristic. Its flowchart is shown in Figure 3. The Expand operation adds a new vertex into a subgraph and updates a measurement of the correlation of this vertex with all vertices in the subgraph. Correspondingly, such measurement of each vertex already in the subgraph is updated as well. The Shrink operation deletes from the subgraph the vertex with the smallest correlation measurement to other vertices, and then update this measurement of each vertex in the subgraph. This search heuristic approach has a complexity of $O(nK)$.

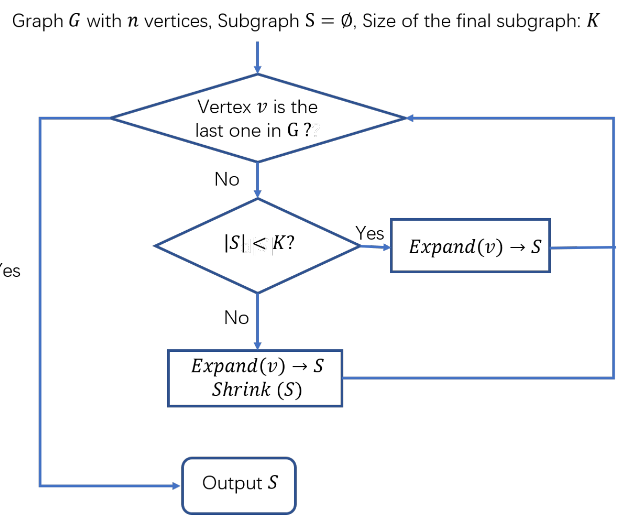

Graph $G$ with $n$ vertices, Subgraph S = ∅, Size of the final subgraph: $K$
Vertex $v$ is the last one in G ?
No
Yes
$|S| < K$?
Yes
$Expand(v) \to S$
No
$Expand(v) \to S$ $Shrink\ (S)$
Output $S$

**Figure 3: Logic of Interchange Search Heuristic**

## 3 Evaluation and Results

Methodologies discussed in the previous section are about the DkS problem. However, the optimization objective of this project is to minimize visualization quality loss. Hence, in experimental implementation, this project modifies these methods to find the minimal or near-minimal objective values. The code for experiments is available in **https://github.com/t07902301/mini-VAS.git**

### 3.1 Experimental Settings

This project adopts Geolife, a GPS trajectory dataset collected by Microsoft Research Asia [12]. The format of each point in this dataset is (longitude,latitude). All the experiments are runned in a AMD EPYC 7302P 16-Core CPU.

### 3.2 Baselines: sampling-based methods

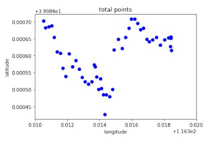
Simple reservoir sampling and stratified sampling are two common practice in big data visualization [10]. Simple reservoir sampling maintains a pool of $k$ items if $k$ points are to be sampled from the whole dataset. Then it traverses the remaining items and replaces these items with those randomly selected from the pool. However, this method tends to select more points from dense regions which will lead to big visualization quality loss. Stratified sampling is an approach to overcome the limitation of simple reservoir sampling. This approach splits points into bins, and selects some points from each bin by simple reservoir sampling. But it is often hard to determine how many bins to generate in stratified sampling.

### 3.3 Experimental Results

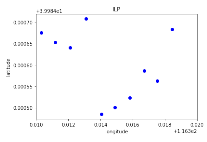
Table 2 shows performance of different algorithms on three **small-scale** datasets. The size of subset selected from these datasets is 10. Stratified Sampling uses 10 bins to divide data. Evaluation metrics in this table are the running time of a method and the objective value of the solution this method generated.

In all these small datasets, BLP is able to find better data reduction solutions than any other algorithms studied here. However, it also takes much longer time to run. Hence, we can infer that BLP is not suitable for visualizing big datasets. As for the $\frac{1}{2}$-approximation algorithm, it runs fast but it cannot find good solutions since its corresponding objective values are the largest among all the algorithms.

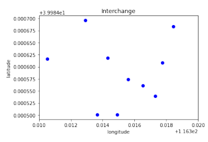
Figure 4 is about plots based on subsets selected by these five methods. The plot of subset generated by BLP reveals more information of the groundtruth compared to other methods. While the plot regarding to the $\frac{1}{2}$-approximation is the most far away from the groundtruth. Hence, the objective value is positively related to the similarity between the subset plot and the groundtruth.



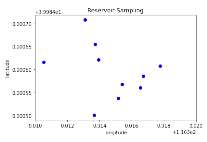
**(a) Groundtruth**



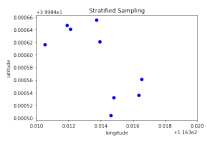
**(b) BLP**



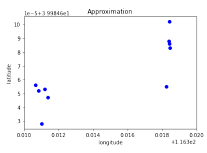
**(c) Interchange**



**(d) Reservoir**



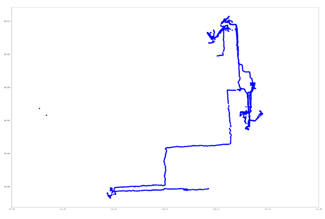
**(e) Stratified**

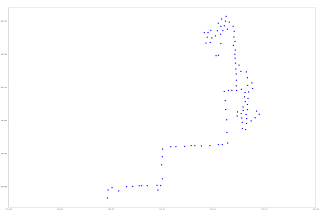


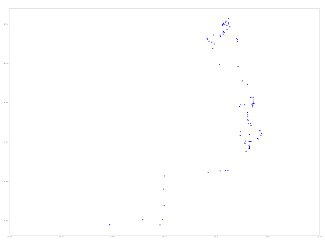
**(f) Approximation**

**Figure 4: Visualizing 10 points from 50-Dataset**

Based on a preliminary experiment on small datasets in the previous subsection, we can consider Interchange Search Heuristic, Reservoir and Stratified sampling methods as promising methods for visualization of **large-scale** datasets. Therefore, we only experiment these three methods on large datasets. A random permutation of dataset of 1M size is performed before running Interchange so that this algorithm has more chance to explore points from this dataset as many as possible. Table 3 displays evaluation results of these methods on three large datasets. Stratified Sampling has 100 bins for data partition in these experiments. Evaluation metrics here are the same as those for small datasets. We have to interrupt Interchange otherwise it takes very long time to stop. But it can still find much better data reduction solutions than two sampling approaches. Figure 5 also demonstrates that the superiority of Interchange over sampling methods in shrinking large datasets for visualization. The plot it generated is the most similar to the groundtruth.


**(a) Groundtruth**


**(b) Interchange**


**(c) Reservoir**

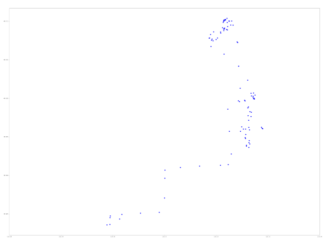

**(d) Stratified**

**Figure 5: Visualization of 100 points from 2000-dataset**

However, if Interchange is given more time to run, it will find solutions with less quality loss. This fact is illustrated in Figure 6. This figure displays the quality decay of subsets with size 800 and 1000 from a dataset with one million points.

## 3.4 Discussion

Designing a data reduction strategy for visualizing large datasets is about a trade-off between the visualization quality and the required time. BLP can find the best data reduction solution, but it takes much time to converge even in small datasets.

**Table 2: Results on Small Datasets with Subset Size = 10**

| Dataset size | BLP | Interchange | Reservoir | Stratified | $\frac{1}{2}$-approximation |
|---|---|---|---|---|---|
| 30 | 1m41s/3.93 | <0.1s/4.13 | <0.1s/7.47 | <0.1s/10.08 | <0.1s/15.87 |
| 40 | 7m24s/1.92 | <0.1s/2.08 | 0.1s/5.1 | 0.1s/5.5 | <0.1s/15.96 |
| 50 | 41m36s/0.73 | <0.1s/1.95 | <0.1s/5.04 | <0.1s/6.14 | <0.1s/16.57 |

**Table 3: Results on Large Datasets**

| Dataset/Subset Size | Interchange | Stratified | Reservoir |
|---|---|---|---|
| 2000/100 | 4.84s/10.7 | <1s/135.01 | <1s/ 171.73 |
| 10K/500 | Interrupted at 6s/1790.87 | <1s/3450.35 | <1s/4433.79 |
| 1M/1000 | Interrupted at 9s/438.79 | 1.5s/778.53 | 1.5s/864.63 |

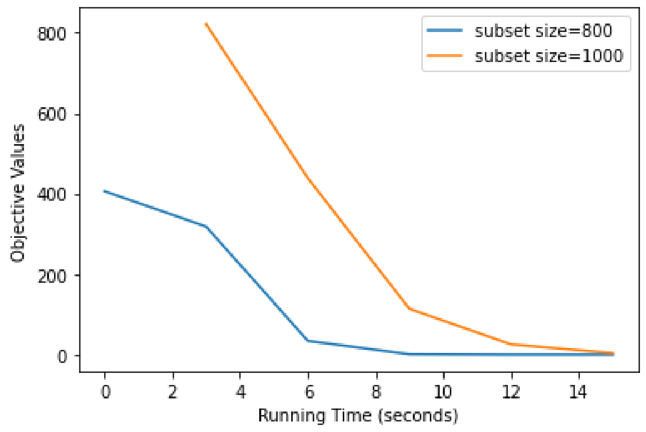


**Figure 6: Running Time V.S. Objective Values**

Sampling methods are fast enough to decide how to reduce data for visualization, however, they often lead to big quality decay. Interchange search heuristic also suffers from huge latency when used in large datasets. But given a tolerably long time span, it can bring us with much better visualization quality than sampling methods.

## 4 Conclusions

This project studies six methodologies to solve a NP-hard combinatorial optimization problem of visualizing large datasets via data reduction. Among these methods, Interchange Search Heuristic achieves the best performance if it is allowed to run in a time span with a tolerable length. However, this search heuristic cannot guarantee optimal solutions, since it usually does not have enough time to look over the whole dataset. Hence, more advanced approaches to solving this optimization problem are needed to be applied and evaluated for large-scale dataset visualization.

## References

[1] [n.d.]. PII: S0167-6377(97)00034-5 | Elsevier Enhanced Reader. https://doi.org/10.1016/S0167-6377(97)00034-5

[2] Moscow Aviation Institute Moscow Russia Andrew Makhorin, Department for Applied Informatics. 2022. GNU Linear Programming Kit. https://www.cs.unb.ca/~bremner/docs/glpk/glpk.pdf.

[3] Aditya Bhaskara, Moses Charikar, Eden Chlamtac, Uriel Feige, and Aravindan Vijayaraghavan. 2010. Detecting high log-densities: an $O$ ( $n^{¼}$ ) approximation for densest $k$ -subgraph. In *Proceedings of the 42nd ACM symposium on Theory of computing - STOC '10*. ACM Press, Cambridge, Massachusetts, USA, 201. https://doi.org/10.1145/1806689.1806719

[4] Alain Billionnet. 2005. Different Formulations for Solving the Heaviest $K$ -Subgraph Problem. *INFOR: Information Systems and Operational Research* 43, 3 (Aug. 2005), 171–186. https://doi.org/10.1080/03155986.2005.11732724

[5] Danyel Fisher, Igor Popov, Steven Drucker, and m.c schraefel. 2012. Trust me, i'm partially right: incremental visualization lets analysts explore large datasets faster. In *Proceedings of the SIGCHI Conference on human factors in computing systems (CHI '12)*. ACM, 1673–1682.

[6] Aritra Konar and Nicholas D. Sidiropoulos. 2021. Exploring the Subgraph Density-Size Trade-off via the Lov\'asz Extension. *arXiv:2102.03434 [cs]* (Feb. 2021). http://arxiv.org/abs/2102.03434 arXiv: 2102.03434.

[7] Zhicheng Liu, Biye Jiang, and Jeffrey Heer. 2013. imMens: Real-time Visual Querying of Big Data. *Computer graphics forum* 32, 3pt4 (2013), 421–430.

[8] Elder Magalhães Macambira. [n.d.]. An Application of Tabu Search Heuristic for the Maximum Edge-Weighted Subgraph Problem. ([n. d.]), 16.

[9] Joachim Dahl Martin S. Andersen and Lieven Vandenberghe. 2022. CVXOPT - Python Software for Convex Optimization. https://cvxopt.org/.

[10] Yongjoo Park, Michael Cafarella, and Barzan Mozafari. 2015. Visualization-Aware Sampling for Very Large Databases. (Oct. 2015). http://arxiv.org/abs/1510.03921 _eprint: 1510.03921.

[11] Renata Sotirov. 2019. On solving the densest $k$-subgraph problem on large graphs. *arXiv:1901.06344 [math]* (Jan. 2019). http://arxiv.org/abs/1901.06344 arXiv: 1901.06344.

[12] Yu Zheng. 2010. GPS Trajectories with transportation mode labels. https://www.microsoft.com/en-us/research/publication/gps-trajectories-with-transportation-mode-labels/ Edition: GeoLife project-Microsoft Research Asia.