An efficient nd-point data structure for querying flood risk

Liu, H.; Van Oosterom, P.; Mao, B.; Meijers, M.; Thompson, R.

DOI
10.5194/isprs-archives-XLIII-B4-2021-367-2021

Publication date
2021

Document Version
Final published version

Published in
International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives

AN EFFICIENT ND-POINT DATA STRUCTURE FOR QUERYING FLOOD RISK

H. Liu[a,*], P. Van Oosterom[a], B. Mao[b], M. Meijers[a], R. Thompson[c]

[a] Faculty of Architecture and the Built Environment, Delft University of Technology, Delft, the Netherlands
(H.Liu-6, P.J.M.vanOosterom, B.M.Meijers)@tudelft.nl
[b] Changjiang River Scientific Research Institute, Wuhan, China
bingm@whu.edu.cn
[c] 39 Salstone Street Kangaroo Point, Brisbane, Australia
rodnmaria@gmail.com

KEY WORDS: Flood mapping, nD point clouds, Database, Space filling curve, Hydrology.

ABSTRACT:

Governments use flood maps for city planning and disaster management to protect people and assets. Flood risk mapping projects carried out for these purposes generate a huge amount of modelling results. Previously, data submitted are highly condensed products such as typical flood inundation maps and tables for loss analysis. Original modelling results recording critical flood evolution processes are overlooked due to cumbersome management and analysis. This certainly has drawbacks: the 'static' maps impart few details about the flood; also, the data fails to address new requirements. This significantly confines the use of flood maps. Recent development of point cloud databases provides an opportunity to manage the whole set of modelling results. The databases can efficiently support all kinds of flood risk queries at finer scales. Using a case study from China, this paper demonstrates how a novel nD-PointCloud structure, HistSFC, improves flood risk querying. The result indicates that compared with conventional database solutions, HistSFC holds superior performance and better scalability. Besides, the specific optimizations made on HistSFC can facilitate the process further. All these indicate a promising solution for the next generation of flood maps.

1. INTRODUCTION

To prepare for potential future flood disaster (Figure 1), many countries have conducted flood mapping projects in past decades (Merz et al., 2007, Cheng, 2005, Tran et al., 2009). For instance, in USA, the Federal Emergency Management Agency (FEMA) generated digital flood maps for most of the U.S. population (Council et al., 2009). The European Water Directors also established the European Exchange Circle on Flood Mapping (EXCIMAP) to gather all existing experiences in Europe on flood mapping to improve related practices. The goals include land-use planning and land management, watershed management, hazard assessment on local level, emergency planning and management, and insurance (Van Alphen et al., 2009).

Figure 1. Breach of a dike in Japan in 2019. Image source: The Japan News

China has implemented the national flood risk mapping projects for key areas during 2013 to 2015. The mapping process mainly includes two parts. The first part concerns running a 1D and 2D coupled hydrodynamic model to compute water depth, flow speed and direction at different time steps, given a specific breach case. The result is stored in a 2D grid covering the modelling basin, which is then used for making various maps such as the maximum inundation map and inundation duration map. Also, combined with social economic data, potential loss tables are computed. The water ministry collects these final products, and plans to use them for decision making. However, they omit a large part of original modelling result which is cumbersome to manage, analyze and present. This certainly has drawbacks:

1. since the products are 'static' (Figure 2), no more details can be derived;
2. the map fails to address new types of queries, which significantly confines the use of flood maps;
3. many small regions which can alleviate the risk of larger flood storage and retention areas are not modelled, which limits analysis when a real flood happens;
4. for the regions simulated, input breach cases are not enough, because the real breach may be somewhere else;
5. due to limited computing power, spatial and temporal resolution is confined to coarse scales;
6. frequent update due to land use change (Merz et al., 2007, Cheng, 2005) is impossible.

In fact, any specific flood maps can be expressed by SQL queries. For example, the inundation extent is achieved by selecting all the grid cells with water depth larger than 0, while the arrival time map is created by selecting cells at different time

* Corresponding author

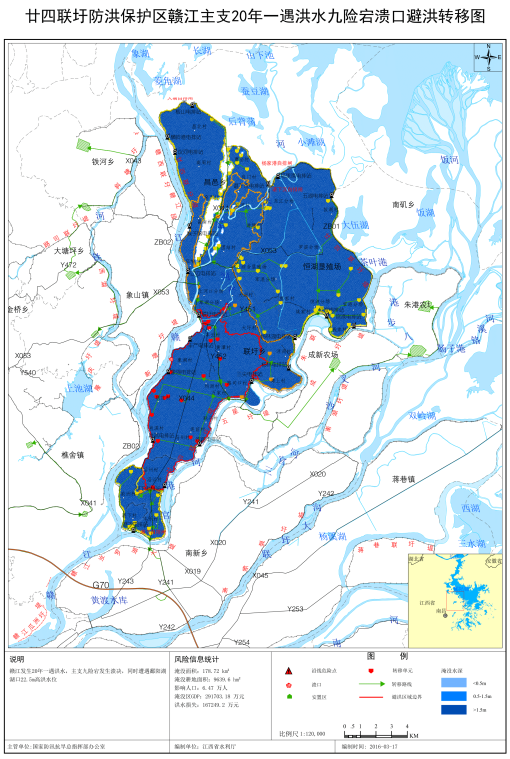

Figure 2. A typical flood evacuation map in China

steps of flooding. In addition, new queries such as flood situation around certain objects can also be resolved. Hence, to address issues listed above, developing an efficient database for ad-hoc queries is imperative.

The issue then becomes which database technology to use. Current solutions for flood risk analysis are mainly based on relational databases, either standalone as flat tables (Brocca et al., 2013), or encapsulated in GIS software (Forkuo et al., 2011, Wang et al., 2010). However, according to practical experience and previous studies in geo-applications (Stonebraker et al., 2013, Van Oosterom et al., 2015), indexing flat tables performs inefficiently faced with large multidimensional data. One major reason is the huge size of the extra indexing structure which makes it cumbersome to use. GIS software mainly focuses on processing and analyzing small datasets, and lacks mechanism to handle huge data. Another aspect to consider is that current flood models mainly use irregular grids, e.g., triangular networks for computation. So, the multidimensional array databases such as SciDB and Rasdaman are inapplicable, as they focus on regular grids (e.g., the raster) (Liu et al., 2018). Alternatively, point cloud databases can be an option:

- we can directly use the point cloud model to manage the flood modelling result. Each point (i.e., the centre of a cell) has breach case ID, X, Y, Z, time, depth, velocity and direction dimensions;
- recent HistSFC framework which is aimed at efficiently managing and querying massive nD point clouds is developed and verified with applications on LiDAR point clouds (Liu et al., 2020). It is implemented in a database, with nD-histograms and irregular querying techniques to solve more complex queries;
- flood risk queries can be performed by using such a database, where different maps can be derived from queries. Besides, the database can address other potential queries, such as evaluating water depth along a road or the risk to a house considering all breach cases.

This paper investigates the applicability of the nD-PointCloud structure, HistSFC for querying flood risk, and presents its advantage in terms of functionality and efficiency. The rest of the paper is organized as follows: Section 2 discusses the state-of-the-art HistSFC, including novel optimizations we made for flood queries. Section 3 tests the performance of HistSFC, and demonstrates its superiority over conventional solutions based on a case study in China. In the end, the paper concludes with summary of main results and future work in Section 4.

2. HISTSFC

In point cloud data, each point contains multiple dimensions. In terms of data management, we distinguish two types of dimensions. *Organizing dimensions* are used to cluster and index the data, e.g., X/Y/Z/T. The other *property dimensions* are affiliated, such as color, intensity and classification, which are not frequently used in the SQL WHERE clause. These two types of dimensions are not fixed, and may be varied depending on applications.

Another key concept is Space Filling Curve (SFC). It is an advanced technique to cluster and access data, and has been adapted and improved for multidimensional point data management (Wang and Shan, 2005, Zhang et al., 2014). Among all SFCs, the Morton curve is commonly studied and practiced due to the simplicity of mapping functions (Morton, 1966). It is based on interleaving the bits from the coordinates.

HistSFC (Liu et al., 2020) utilizes Morton curve to convert all organizing dimensions into a Morton key for storage. For example, given a point with coordinates (3,2), its binary representation is (11, 10). By interleaving these bits, the Morton key can be derived which is 1101, i.e., 13 as a decimal number. In such a way, all nD-points can be converted to 1D Morton keys. Then, one-dimensional indexing structure such as the B+-tree can be used to retrieve the keys, to address a query. The remaining work is to map the query window to ranges of Morton keys for selection. Section 2.1 specifies this process.

2.1 Primary settings

As is mentioned, each nD-point is encoded as a full resolution Morton key, a combination of all organizing dimensions. Property dimensions are attached to the key. Such a full resolution key can be decoded directly to the original coordinates. Figure 3 presents the workflow of HistSFC. For storage, after computing SFC keys, HistSFC adopts the Index-Organized Tables (IOT) (Oracle, 2013a) to manage them. Oracle has implemented the B+-tree structure in IOT: the key and other property dimensions are stored in leaf nodes present in a flat table, while the internal B+-tree uses the SFC keys to organize and index the storage. In this way, the indexing structure is integrated with the storage, which lead to a very compact structure.

Before executing queries, we first build the HistTree. HistTree is devised to represent the data distribution to avoid excessive range generation in sparse areas, in the presence of a skewed point distribution. As Figure 4 shows, HistTree counts the

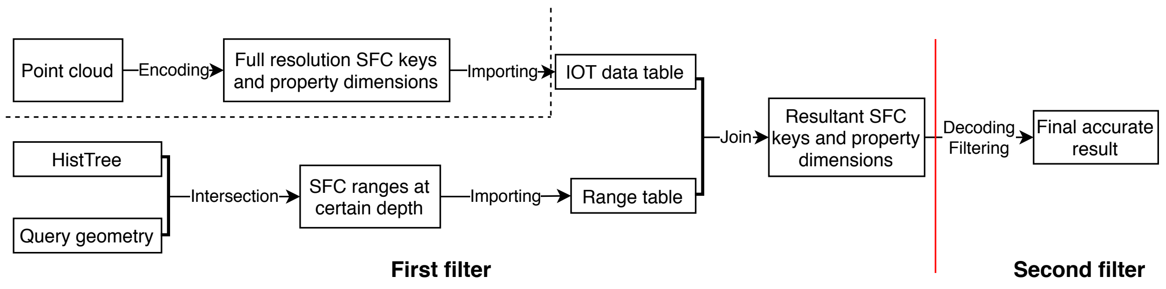

Figure 3. The loading and querying procedure of HistSFC

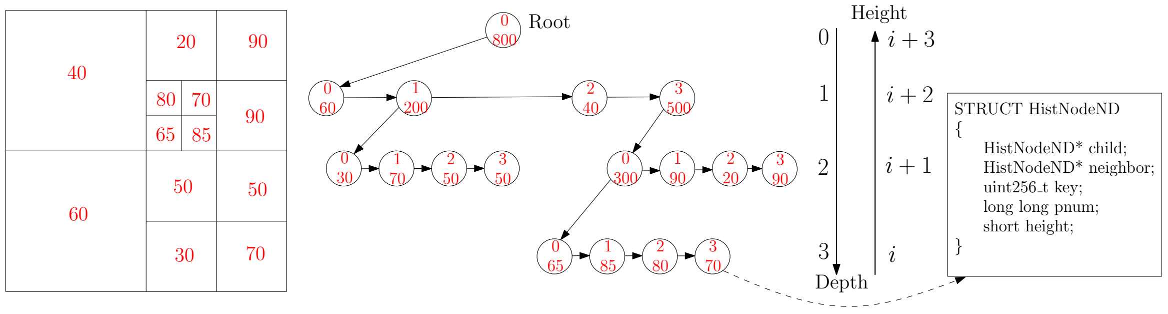

Figure 4. A 2D HistTree example, where the threshold is 100; left: point counting, middle: pointer structure of HistTree, with each node storing a SFC key and number of points, right: structure of a HistTree node

points in each SFC node at different level. If the number exceeds the threshold of the tree, the node will be partitioned into nodes at a lower level. A height field is used in a HistTree node to distinguish different nodes, because branch nodes at different levels may possess identical keys. A HistTree node actually represents the MBB of a quadrant, but it contains neither points nor pointers to points. Thus, HistTree is a compact structure which can be stored in a flat table.

When querying (Figure 3), HistSFC employs HistTree to map the query window to 1D SFC ranges. Starting from the root node, the extent of each HistTree node can be computed using its height and the key. Then, by performing intersection between branch nodes and the query window iteratively, HistSFC retrieves all intersected leaf nodes, and abandons non-overlapping nodes. Some of the nodes retrieved locate totally inside the query window. HistSFC directly exports ranges represented by them. The other resultant leaf nodes fall on the boundary of the query window. These can be further refined based on a recursive fixed decomposition. That is, halving every dimension to build child nodes in each iteration. The process stops when the number of ranges reaches the maximum we set. The IOT strategy then uses the B+-tree to select keys within all these ranges. The returned keys (points) are then applied to the second filter to select the required set of points (Figure 3).

The theoretical querying time of HistSFC is as follows:

$$T = r \cdot t_{pre} + \left(\sum_{i=1}^{r} \left\lceil \frac{k_i}{B} \right\rceil \right) \cdot t_{io} + k' \cdot t_{post} \qquad (1)$$

where B = page capacity in the number of points
r = number of ranges generated
k_i = number of points inside a specific range
k' = number of points returned by the first filter
$\sum_{i=1}^{r} k_i = k'$

In Equation 1, t_{pre} elaborates the computing time and scanning time of one range for joining with the SFC point records. The middle term indicates the I/O cost, and t_{io} refers to the time cost to load one page from disk. In fact, the middle term is an upper limit of I/O operations, as different ranges might cover the same disk page which has been counted more than once. In the last term, t_{post} refers to the post processing after retrieving one SFC record, including decoding, point-in-window computation and exporting. Parallelism can be implemented for post-processing in a straightforward way (Section 2.2.3). Basically, we divide SFC ranges among p processors, each of which will get part of k'. So, the final decoding will cost about $\frac{k'}{p} \cdot t_{post}$ time, given a balanced workload.

2.2 Optimizations

The first filter leverages the B+-tree for querying, which is efficient. However, due to the constraint of maximum number of ranges, it may use larger coarse ranges and thus return additional false positive points which can only be filtered out by the second filter. In fact, flood risk queries often result in large output, e.g., to support whole basin management and statistical analysis. Thus, we optimize HistSFC in the following.

2.2.1 Range refinement In current settings, boundary leaf nodes selected will be recursively decomposed in a breadth-first way to intersect the query window, to derive child nodes (Figure 5). So, HistSFC can filter out more false positive points. This strategy assumes all leaf nodes are equally important for decomposition, which may not be appropriate.

In Figure 5, some SFC cells (i.e., nodes) intersect with the query window by a large proportion. Thus, most points inside these cells are within the query window as well. In contrast, other boundary cells, i.e., P_1, P_2 and P_3, which intersect the query window by a small portion mainly contain false positives. These cells should get priority for the refinement such that a decomposition becomes more effective. Consequently, we can optimize the nodes' partitioning process to refine the ranges, by considering the intersection ratio which equals the volume of intersection divided by the volume of node.

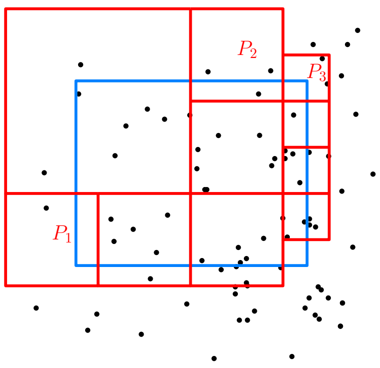

Figure 5. A 2D query sample: leaf nodes selected (red) in the HistTree to match the query window (blue)

Specifically, we compute the intersection ratio in the first filter and rank the intersected leaf nodes by the estimated number of false positive points (Equation 2). The leaf nodes with more outliers will be decomposed first. After one decomposition, we assume points are allocated evenly to different child nodes. Then, we add the child nodes into the original pool of leaf nodes, and rank them again for further decomposition.

$$\tilde{fpp} = (1 - R) \times N_c \tag{2}$$

where $\tilde{fpp}$ = estimated number of false positive points in a boundary node
R = intersection ratio
N_c = number of points in the boundary node

2.2.2 Polytope querying HistSFC is not restricted to rectangular search regions, but can also deal with irregular geometry querying, such as an nD sphere or triangle. The recursive partition of the SFC node to match the query geometry still applies. However, we need to overwrite the intersection module for different types of geometry. To make the solution more generic, we use half-spaces composing a convex polytope to approximate the real query geometry, and develop a generic sweep algorithm (Thompson et al., 2020) for intersection computation.

The sweep algorithm first identifies the enter and exit (one of the vertices) of a SFC node with respect to a half-space (, $\sum_{i=1}^{n} a_i x_i - b \geq 0$). For each dimension, if the corresponding $a_i \geq 0$, then the enter takes the lowest value at that dimension. Otherwise, it adopts the highest value. The exit is the opposite of the enter (Figure 6). As an illustration, Figure 6 indicates how the algorithm determines whether a node intersects a polytope composed by H1 and H2:

- Case 1: if the enter is inside the half-space, then the node is fully inside the half-space, and therefore is fully inside the polytope;
- Case 2: if the exit is outside the half-space, then the node is totally outside the polytope;
- Case 3: if the enter is outside the half-space, but the exist is within the half-space, then the node is cut by the half-space boundary (, e.g., for both H1 and H2). The node intersects the polytope;

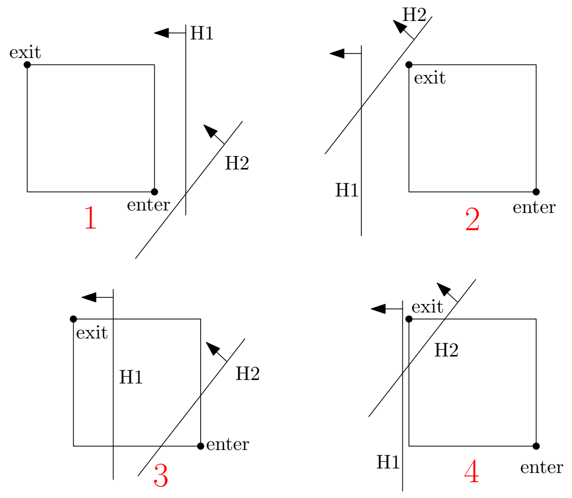

Figure 6. Four cases of intersection between a SFC node and half-spaces in the 2D space

- Case 4: if the enter is outside the half-space, but the exit is inside, then the node intersects the half-space (H2). However, if the exist is outside another half-space (H1), then the node is totally outside the polytope.

For each node, HistSFC examines all half-spaces. If intersection happens, HistSFC decomposes the node to its children to check again. During this process, HistSFC records the half-spaces totally containing the node so that no more examination will be conducted for its children. Till the maximum of ranges generated, the algorithm exports all intersected nodes.

2.2.3 Parallel decoding The previous optimizations are aimed at decreasing the false positive keys returned by the first filter. However, the decoding process can still be time consuming if large amount of keys have to be processed anyway. To address this issue, we adopt the parallel technique for decoding. A straightforward way is to evenly distribute the ranges to different processors so that each processor performs a query and decode the result. This can be easily implemented as all processors adopt the same function.

3. USE CASE STUDY

The research area is Niansi Levee, located at Jiangxi province, China (Figure 7). To its northwest is the Poyang Lake. The total area is about 183 km^2. Niansi Levee is one of the key areas that are modelled in the national flood mapping project.

3.1 Data

The hydrodynamic model consists of a 1D channel model, and a coupled 2D flood routing model in the basin. The channel model provides extreme flow for the flood simulation in the 2D grid which totally contains 59,680 triangular cells. The model computes water depth, velocity and flow direction. We modelled 8 cases: 4 locations of breach, combined with extreme rainfall with a return period of 20 and 50 years. Each case simulates 720 steps (corresponding to a 30-min resolution). So in total, we get 59,680 × 720 × 8 = 343,756,800 points, in an 8D space composed by case ID, X, Y, Z, time, depth, velocity and direction.

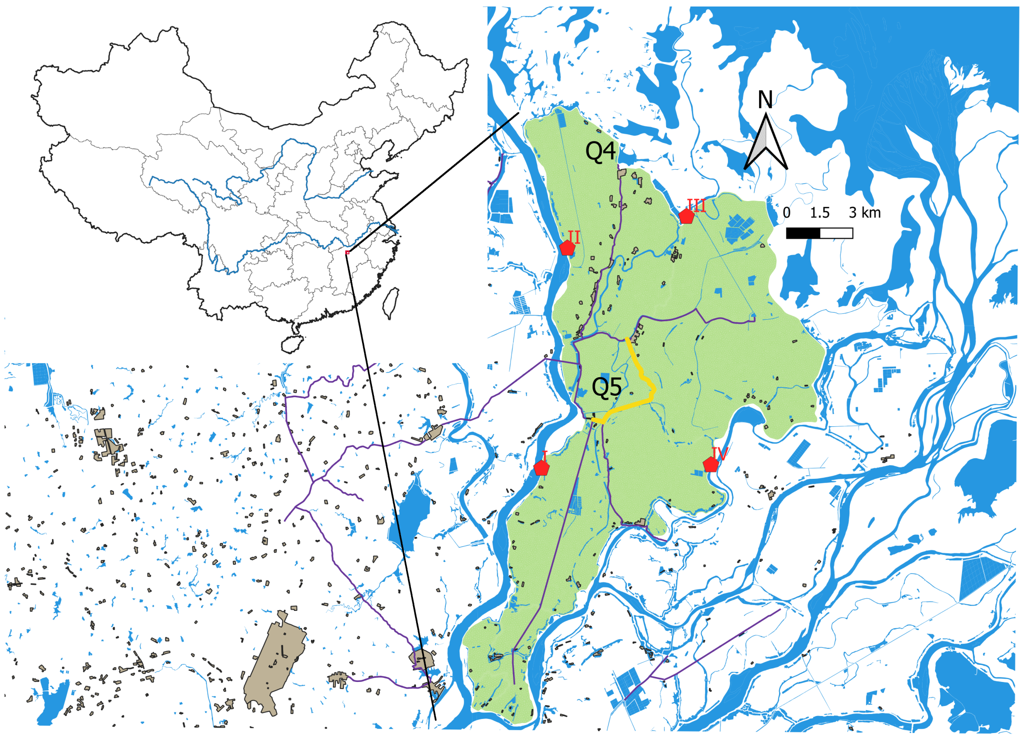

Figure 7. Niansi Levee (green zone), with locations of simulated breach in red

3.2 Queries

According to the experience acquired during the project, as well as potential need, we devised 6 queries for testing (Table 1). All locations are depicted in Figure 7, e.g., Q4 and Q5.

QID	Analysis
Q1	The area that is flooded with depth greater than 3 m, when dyke I bursts
Q2	The area that is flooded in 24 hours, when dyke IV bursts
Q3	The maximum inundation area when dyke III bursts
Q4	Risk of several houses (depth > 0) when dyke II or III bursts
Q5	Risk to a county road (velocity ≥ 0.5) considering all possible bursts
Q6	The dangerous area evaluated by human instability (depth × velocity ≥ 2), when dyke I bursts

Table 1. Queries used for benchmarking

Q4 selects all points around 4 houses that have been flooded. The rectangular area is about 1.5 km^2. Q5 uses an irregular geometry (i.e., the road) to query. The result of Q5 is 93 8D points, presented as 6 distinct spatial points at different time steps in Figure 8. The points indicate the vulnerable parts of the road which need enhancement. During flood evacuation, people should also avoid these locations. Q6 uses the product of flow velocity and depth to quantify the human instability in flowing water (Jonkman and Penning-Rowsell, 2008). We choose 2 m^2/s, a rather safe level, as the threshold.

3.3 Implementation

The test is conducted on a 'testbed' server: a HP DL380p Gen8 server with 2 × 8-core Intel Xeon processors, E5-2690 at 2.9 GHz, 138 GB of main memory, a RHEL6 operating system. The disk storage is a 41 TB SATA 7200 rpm in RAID6 configuration.

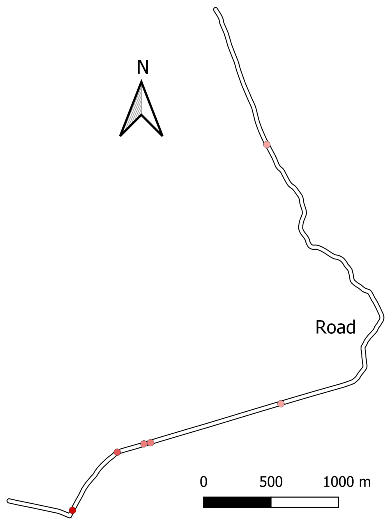

Figure 8. Result of Q5: the red points on the road. The color becomes stronger when the maximum velocity is larger

Among the 8 dimensions of the data, flow direction is seldomly used for ad-hoc analysis, compared with other dimensions. So, we set it as the property dimension when building HistSFC, while encode the other 7 dimensions into the Morton key stored in IOT. We use 1000 as the threshold to build the HistTree. As a comparison, we also build a flat table which stores each dimension as an individual column. Although this approach performs inefficiently, it serves as a baseline to compare with. Solutions provided by major spatial databases fail to address nD-point management. For instance, Oracle SDO_PC (Oracle, 2013b) builds point blocks based on a 2D organization, while PostGIS can maximally support a 4D-point geometric type.

To explore the scalability, both approaches divide the whole dataset into 4 stores, containing 1, 2, 4 and 8 cases respectively. The size of HistSFC store with 8 cases is 12.88 GB, and that of the HistTree is 84 MB on the disk. The flat model occupies 15.4 GB disk space.

3.4 Benchmark tests

For all queries, HistSFC utilizes 16 processors to decode Morton keys. Besides, it employs 7 half-spaces to approximate the shape of Q6, for querying (Figure 9). More details about implementing data storage and queries can be found in the Appendix.

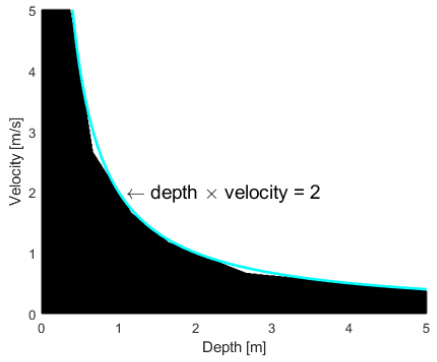

Figure 9. Polytope querying of Q6

Figure 10 shows all testing results. It indicates that for all queries, compared to the flat table approach, HistSFC holds better scalability, following a constant trend. In most cases, the time cost is much lower than the flat table solution. An exception is Q6, where HistSFC performs significantly worse. This is because current parallel implementation evenly distributes the ranges to different processors. As each range contains different number of points due to skewed data distribution, the actual workload can be unbalanced among processors. In this case, we found a processor undertakes 72% of the whole workload.

More specifically, Q2 takes less time on the whole, compared with Q1 and Q3. This is because it selects only 925,691 points out of the total 343,756,800 points, while Q1 and Q3 export 26,484,215 and 32,183,314 points, respectively. Q4 is also very selective, returning 170,417 points. It first does a spatial window selection (, i.e., a rectangular area with houses), and then checks the flood information which is indicated by a positive depth value. Unlike Q4, Q5 utilizes a long and narrow 2D polygon for the spatial selection, consisting of 628 edges. This, however, does not introduce significant overhead compared with Q4. Q6 uses the polytope querying technique (Figue 9), which leads to a resultant 30,937 point set, slightly larger than the correct answer 28,351. The false positive rate is below 1%. For all other queries, the two approaches return the same results.

3.5 Discussion

Flood analysis is frequently performed globally, such as Q1, Q2 and Q3. Therefore, a large output may always be encountered. To perform such queries efficiently, low I/O operations and parallel post-processing become very crucial, indicated by Equation 1. Both processes are related to the data distribution: if data is severely skewed in the nD space, some ranges generated will contain lots of false positives, which significantly increases I/O; besides, the uneven distribution will cause unbalanced workload among processors (Figure 10(f)). It is possible to perform the first filter on a single thread, and then distribute the keys selected for decoding. However, additional memory and time cost will become an issue. A direct solution is to estimate the number of points in the final ranges, e.g., based on the length of the range. Then, each processor gets the job allocated according to the points estimated.

Besides, the dimensionality is a variable. The test dataset contains only one levee. In fact, to defend flood, groups of levees are built and used. Consequently, a levee ID may also get involved for an integrated database. Encoding too many dimensions into the SFC key is not sensible, as this will significantly decrease the accuracy of the first filter (Liu et al., 2020). The consequent I/O cost will become huge. Hence, it is suggested that before using HistSFC, a comprehensive analysis of applications and data should be conducted to determine the proper organizing dimensions. For example, this research does not consider using flow direction for data organization, as it is not queried often.

4. CONCLUSIONS AND FUTURE WORK

This paper has investigated the possibility of using an nD-PointCloud structure — HistSFC — for the next generation of flood maps, to fulfill increasing needs of different stakeholders. This research also developed critical optimizations on HistSFC to resolve flood issues. Then, a benchmark test is performed to evaluate HistSFC's performance in practice. The result indicates that the optimized HistSFC can process various flood queries very efficiently. The scalability remains stable as the input size increases. Although the paper only presents 6 queries, HistSFC can address more complex nD queries in an analogous way. The polytope querying technique is also generic: in additional to the polygonal spatial queries, more query geometries formed in other physical dimensions can be addressed. Besides, with the parallel post-processing, HistSFC can be improved further using state-of-the-art hardware, e.g., not only powerful workstations, but also cloud computing platforms. We can additionally parallelize the range computing process to enhance HistSFC.

Point cloud databases enable dynamic analysis on large-scale flood simulations. However, the SQL interface is less friendly and requires high level of expertise to master. So, in the future, we plan to develop a Graphical User Interface (GUI) as the front-end, to drive various queries and analysis. User requirements must be collected for devising such an integrated flood mapping system. Besides, the GUI should support efficient rendering, as large-scale point visualization may become the routine. This reversely calls for new design in the data structure, such as embedding Level of Detail (LoD). This verifies the advantage of HistSFC which can just treats LoD as another organizing dimension. Specific LoD that is applicable to flood analysis should be researched and developed in the next step.

An nD-point data structure managing flood modelling results is innovative, as the major use of high dimensional points is still in the surveying domain and initial stage of applications. However, either a physical point or a simulated point functions as the carrier of data. The nD point cloud model is the core. From this perspective, we can extend the scope of applications, e.g., wind modelling and mechanical analysis of materials.

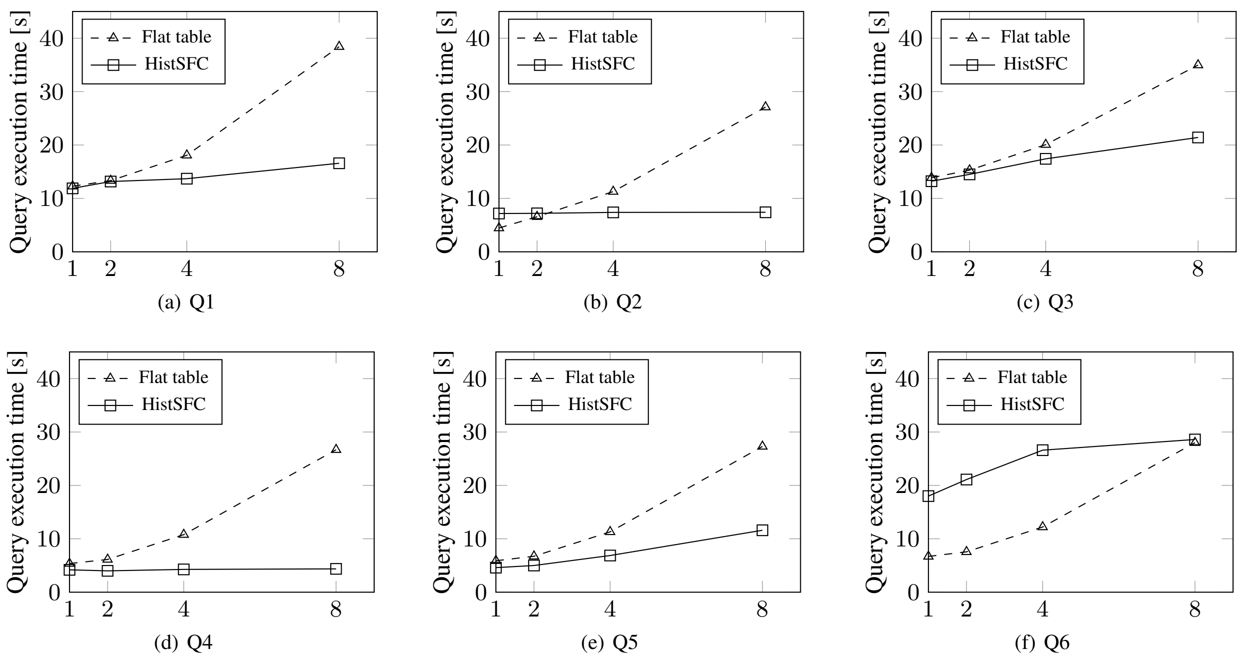

Figure 10. Performance of querying, where X axis represents the number of cases included in the specific data store

ACKNOWLEDGEMENTS

The authors would like to thank the Chinese Scholarship Council (CSC) and Fugro company for supporting this research.

REFERENCES

Brocca, L., Liersch, S., Melone, F., Moramarco, T., Volk, M., 2013. Application of a model-based rainfall-runoff database as efficient tool for flood risk management. *Hydrology and Earth System Sciences*, 17(8), 3159–3169.

Cheng, X., 2005. Basic thinking of pushing flood hazard mapping in China. *China Water Resources*, 17, 11–13.

Council, N. R. et al., 2009. *Mapping the zone: Improving flood map accuracy*. National Academies Press.

Forkuo, E. K. et al., 2011. Flood hazard mapping using Aster image data with GIS. *International journal of Geomatics and Geosciences*, 1(4), 932–950.

Jonkman, S., Penning-Rowsell, E., 2008. Human instability in flood flows 1. *JAWRA Journal of the American Water Resources Association*, 44(5), 1208–1218.

Liu, H., van Oosterom, P., Meijers, M., Guan, X., Verbree, E., Horhammer, M., 2020. HistSFC: Optimization for nD massive spatial points querying. *International Journal of Database Management Systems (IJDMS)*, 12(3), 7–28.

Liu, H., van Oosterom, P., Tijssen, T., Commandeur, T., Wang, W., 2018. Managing large multidimensional hydrologic datasets: A case study comparing NetCDF and SciDB. *Journal of Hydroinformatics*, 20(5), 1058–1070.

Merz, B., Thieken, A., Gocht, M., 2007. Flood risk mapping at the local scale: concepts and challenges. *Flood risk management in Europe*, Springer, 231–251.

Morton, G. M., 1966. A computer oriented geodetic data base and a new technique in file sequencing.

Oracle, 2013a. Indexes and Index-Organized Tables. `https://docs.oracle.com/database/121/CNCPT/indexiot.htm#CNCPT721`.

Oracle, 2013b. SDO_PC_PKG Package (Point Clouds). `https://docs.oracle.com/database/121/SPATL/sdo_pc_pkg-package-point-clouds.htm#SPATL172`.

Stonebraker, M., Brown, P., Zhang, D., Becla, J., 2013. SciDB: A database management system for applications with complex analytics. *Computing in Science & Engineering*, 15(3), 54–62.

Thompson, R., van Oosterom, P., Meijers, M., 2020. Execution of queries on nd pointclouds using convex polytopes. NCG Symposium.

Tran, P., Shaw, R., Chantry, G., Norton, J., 2009. GIS and local knowledge in disaster management: a case study of flood risk mapping in Viet Nam. *Disasters*, 33(1), 152–169.

Van Alphen, J., Martini, F., Loat, R., Slomp, R., Passchier, R., 2009. Flood risk mapping in Europe, experiences and best practices. *Journal of Flood Risk Management*, 2(4), 285–292.

Van Oosterom, P., Martinez-Rubi, O., Ivanova, M., Horhammer, M., Geringer, D., Ravada, S., Tijssen, T., Kodde, M., Gonçalves, R., 2015. Massive point cloud data management: Design, implementation and execution of a point cloud benchmark. *Computers & Graphics*, 49, 92–125.

Wang, J., Liang, Z., Shi, Y., 2010. Mapping of flood risk of reservoirs using GIS technology. *Journal of Hohai University (Natural Sciences)*, 38(1), 20–25.

Wang, J., Shan, J., 2005. Space filling curve based point clouds index. *Proceedings of the 8th International Conference on GeoComputation*, 551–562.

Zhang, R., Qi, J., Stradling, M., Huang, J., 2014. Towards a painless index for spatial objects. *ACM Transactions on Database Systems (TODS)*, 39(3), 1–42.

APPENDIX

Table creation

Flat table

```
CREATE TABLE NS_FLAT (CaseID NUMBER, X
NUMBER, Y NUMBER, Z NUMBER, T NUMBER,
Depth NUMBER, V NUMBER, Di NUMBER);
```

HistSFC

First, we create a staging table:

```
CREATE TABLE NS_SFC (SFC NUMBER, Di
NUMBER);
```

Then, we load data into NS_SFC using SQL*LOADER. After loading, we build the IOT table:

```
CREATE TABLE NS_IOT (SFC, Di, CONSTRAINT
NS_IDX PRIMARY KEY(SFC)) ORGANIZATION
INDEX AS SELECT * FROM NS_SFC;
```

HistTree

```
CREATE TABLE HistTree_NS (ID NUMBER, SFC
NUMBER, PNUM NUMBER, CNUM NUMBER, Height
NUMBER, Child NUMBER, Neighbor NUMBER);
```

Query SQL

Flat table

All the breaches only consider rainfall with a return period of 20 years. In Q5, the road is selected by using geometric operators in the Boost C++ library.

- Q1 `SELECT * FROM NS_FLAT WHERE CaseID=1 AND Depth>=3;`
- Q2 `SELECT * FROM NS_FLAT WHERE CaseID=4 AND Depth>0 AND T<=48;`
- Q3 `SELECT * FROM NS_FLAT WHERE CaseID=3 AND Depth>0;`
- Q4 `SELECT * FROM NS_FLAT WHERE CaseID BETWEEN 2 AND 3 AND X BETWEEN Xmin AND Xmax AND Y BETWEEN Ymin AND Ymax AND Depth>0;`
- Q5 `SELECT * FROM NS_FLAT WHERE CaseID BETWEEN 1 AND 4 AND V>=0.5;`
- Q6 `SELECT * FROM NS_FLAT WHERE CaseID=1 AND Depth*V>=2;`

HistSFC

Firstly, HistSFC converts all query geometries into ranges which are stored in a temporary range table: `CREATE TABLE RANGE_PACKS (LOWER NUMBER, UPPER NUMBER).` Then, it executes all queries as follows,

```
SELECT /*+ USE_NL (t r) */ t.* FROM NS_IOT
t, RANGE_PACKS r WHERE t.SFC BETWEEN
r.LOWER AND r.UPPER;
```