HierGP: Hierarchical Grid Partitioning for Scalable Geospatial Data Analytics

OLUFUNSO OJE, Washington State University, USA
TASHI STIREWALT, Washington State University, USA
OFER AMRAM, Washington State University, USA
PERRY HYSTAD, Oregon State University, USA
SOLMAZ AMIRI, Washington State University, USA
ASSEFAW GEBREMEDHIN, Washington State University, USA

Application domains such as environmental health science, climate science, and geosciences—where the relationship between humans and the environment is studied—are constantly evolving and require innovative approaches in geospatial data analysis. Recent technological advancements have led to the proliferation of high-granularity geospatial data, enabling such domains but posing major challenges in managing vast datasets that have high spatiotemporal similarities. We introduce the Hierarchical Grid Partitioning (HierGP) framework to address this issue. Unlike conventional discrete global grid systems, HierGP dynamically adapts to the data's inherent characteristics. At the core of our framework is the Map Point Reduction (MPR) algorithm, designed to aggregate and then collapse data points based on user-defined similarity criteria. This effectively reduces data volume while preserving essential information. The reduction process is particularly effective in handling environmental data from extensive geographical regions. We structure the data into a multilevel hierarchy from which a reduced representative dataset can be extracted. We compare the performance of HierGP against several state-of-the-art geospatial indexing algorithms and demonstrate that HierGP outperforms the existing approaches in terms of runtime, memory footprint, and scalability. We illustrate the benefits of the HierGP approach using two representative applications: analysis of over 289 million location samples from a registry of participants and efficient extraction of environmental data from large polygons. While the application demonstration in this work has focused on environmental health, the methodology of the HierGP framework can be extended to explore diverse geospatial analytics domains.

CCS Concepts: • **Information systems** → **Data mining**; **Geographic information systems**; *Global positioning systems*; • **Applied computing** → **Environmental sciences**.

Additional Key Words and Phrases: large datasets, hierarchical partitioning, geospatial analytics, environmental health, Landsat imagery analysis

ACM Reference Format:
Olufunso Oje, Tashi Stirewalt, Ofer Amram, Perry Hystad, Solmaz Amiri, and Assefaw Gebremedhin. 2024. HierGP: Hierarchical Grid Partitioning for Scalable Geospatial Data Analytics. *ACM Trans. Spatial Algorithms Syst.* 1, 1, Article 1 (January 2024), 20 pages. https://doi.org/10.1145/3699511

Authors' addresses: Olufunso Oje, olufunso.oje@wsu.edu, Washington State University, Pullman, WA, USA; Tashi Stirewalt, tashi.stirewalt@wsu.edu, Washington State University, Pullman, WA, USA; Ofer Amram, ofer.amram@wsu.edu, Washington State University, Pullman, WA, USA; Perry Hystad, Perry.Hystad@oregonstate.edu, Oregon State University, Corvallis, OR, USA; Solmaz Amiri, solmaz.amiri@wsu.edu, Washington State University, Pullman, WA, USA; Assefaw Gebremedhin, assefaw.gebremedhin@wsu.edu, Washington State University, Pullman, WA, USA.

Manuscript submitted to ACM

1 INTRODUCTION

Advances in computational technology, the availability of low-cost sensors, the growth of the Internet of Things, and remotely sensed geospatial data provide exciting new opportunities to understand complex human-environment interactions. For example, with the heightened occurrence of extreme weather events and a rapidly changing climate [5], there is an acute need to understand the impact of these events on human behavior and health. This is particularly true given that extreme weather events tend to cover large geographic areas and can last several days or weeks [4]. At the same time, the increased availability of spatial-temporal data on human movement provides scientists with the ability to more accurately and efficiently assess environmental exposure over larger geographic areas, assess relationships with behaviors and health outcomes, and potentially intervene in near-real time [10].

For example, our research team uses Google location history data to assess locational smartphone data collected daily over eight years from over 350 study participants in the Washington State Twin Registry. The study aims to understand human behavior and built-environment measures associated with health. Some environmental exposure variables [10], such as temperature, are extracted from satellite data and are calculated for a specific location and time. For over 350 participants, we have collected over 289 million spatiotemporal data points. Future research will expand data collection by upwards of 10,000 participants (>8 billion spatiotemporal records), which presents a technical challenge on multiple levels. In developing data mining and analytics methods for such a dataset, one should exploit the fact that many observations are highly spatially related and contain essentially redundant information. Specifically, the dataset likely contains many observations that are highly similar in both the measurement of interest and **Coordinate Reference System** (**CRS**) values, which should be addressed. A solution to eliminating redundancy of this kind is to represent such a dataset by drastically fewer unique *CRS* points, a process we refer to as **Map Point Reduction** (**MPR**).

In this work, we developed an *MPR* algorithm to create a hierarchical data reduction framework called HierGP. We use this framework to develop heuristics designed to satisfy arbitrary user-specified accuracy criteria. We empirically demonstrate the ability of HierGP to drastically reduce the computational time of extracting environmental data without forfeiting significant sample accuracy.

The *MPR* idea in HierGP can be contrasted with traditional Discrete Global Grid Systems (DGGS) that are often limited by their fixed, predetermined grid structures. While globally consistent, DGGS can lead to inefficiencies in handling data with varying spatial densities and complexities. In contrast, our dynamic approach in HierGP allows users to specify grid sizes and local offsets, enabling adaptation to the unique spatial characteristics of the data. This flexibility is critical in managing the inherent complexities of geospatial datasets, which can vary widely in distribution and density. In a conventional global grid system, the rigid grid structure can result in suboptimal partitioning for data with uneven distribution, leading to either data sparsity or overload in specific grids. By enabling dynamic grid sizing, HierGP mitigates these issues, ensuring more efficient data distribution and retrieval. Furthermore, the linear time complexity ($O(n)$, where n is the number of unique samples) of our *MPR* approach contrasts sharply with the potentially high complexities of other methods. Our method's emphasis on local attention and adaptability enhances data processing efficiency, improves the semantic interpretability of the results, and facilitates more accurate and insightful geospatial mapping analysis.

Summary of Contributions. We summarize our contributions in this work under three categories as follows.

- **Algorithms.** We propose an MPR algorithm to partition large numbers of geospatial points into a significantly smaller set of representative points (Sec 3). We introduce a novel multilevel dynamic hierarchical representation of grid indexing as a basis for the algorithm (Sec 4). The dynamic grid indexing affords a mechanism for

designing heuristics to optimize grid hierarchy levels in large-scale environmental analysis, thereby enabling efficient data processing in compliance with user-defined accuracy threshold (Sec 4.3).

- **Comparison.** We conduct a comparative analysis between HierGP and three existing geospatial algorithms—Uber H3, S2 Geometry Library, and GeoHash—focusing on execution time, memory usage, and scalability. The comparison demonstrates HierGP's superior performance in all three aspects, affirming its suitability for large-scale geospatial tasks (Sec 5).
- **Demonstration.** We showcase HierGP's capability using two illustrative real-world applications. The first application involves analysis of a large-scale dataset containing over 289 million location samples of individuals (Sec 6.1). The second application involves extracting environmental data for large polygons, illustrating dramatic computational efficiency gains in handling traditionally resource-intensive tasks (Sec 6.2).

2 RELATED WORK

We discuss related work in methods for spatial data processing, discrete global grid systems, and quadtree structures, focusing primarily on how the methods compare with our proposed approach.

2.1 Traditional Methods in Spatial Data Processing

Clustering in machine learning and graph partitioning are two broad categories of work that are particularly related to our work. Clustering methods aim to group points using metrics such as distance and density [20]. Extensively studied, clustering methods are the foundation for data analysis efforts across scientific fields [35]. With many variations in approaches, most of the methods rely on centers of points for clustering (e.g., k-Means [19, 20, 30] and K-medoids [23]). The centers of clusters are iteratively updated until a stop criterion is satisfied. Such clustering algorithms include PAM [12], CLARA [13], CLARANS [22]. While these approaches have relatively low time complexities, they are sensitive to outliers and require a hyper-parameter specifying the number of desired clusters beforehand. Various hierarchical-based methods that arrange clusters relative to each other have also been developed (e.g., BIRCH [36], CURE [9], Chameleon [11], etc.). Typically built off existing clustering algorithms (e.g., [16, 26]), these approaches are best suited for datasets with a specific, arbitrary structure. Density-grid-based clustering algorithms have also been proposed as alternatives with varying success [3, 18].

An area of study with many developments relevant to algorithmic reduction is *graph partitioning* [2]. The benefits of heuristics over multilevel graph partitioning have been researched extensively [21]. The multilevel approach efficiently partitions large graphs representing complex data. Following a general reduction process called "coarsening" or "contraction," an approximation of a graph with successively fewer degrees of freedom is made by constructing a hierarchy of successively reduced results. This reduction process terminates when a small enough graph size is reached for use in other partitioning algorithms (geometric partitioning [27], etc.). Following some initial partitioning process, an "uncoarsening" takes each level up the hierarchy in two parts: 1) results are sequentially mapped from coarser to finer levels, and 2) after each mapping, some method is typically used to improve the partition results.

In addition to this generalized form of the multi-step approach, iterative variations (e.g., V-cycle chains) have been proposed to improve the reduction quality and to distribute more of the computational burden to the most reduced levels [25] [33]. Considering modern geographically embedded networks, many recent graph partitioning contributions have made data mining across spatial data increasingly more efficient. Although effective for irregular structures modeled as networks, the graph partitioning approach is too general and thus ill-suited for the regular structure of geospatial datasets.

2.2 Advances in Discrete Global Grid Systems

Another approach is to use Discrete Global Grid Systems (DGGS), where recent advances have opened new avenues for managing and analyzing Earth observation data. Recent advances in DGGS have been comprehensively reviewed [14, 34]. These reviews present DGGS as a unifying framework that tackles the challenges of data storage, processing, and visualization, particularly in the realm of cloud computing [34]. The integration of DGGS with cloud technology is posited as an important stride, offering substantial opportunities yet posing significant challenges, especially in data management, fusion, and grid encoding [14].

Parallel to these developments, explorations into spherical triangular grids [17] and degenerate quadtrees [28] reveal alternative approaches to geospatial data structuring. Additionally, the establishment of a DGGS standard by the Open Geospatial Consortium marks a significant milestone [24]. The standardization is useful for effectively managing and integrating the burgeoning volumes of heterogeneous geospatial data.

The Uber H3 geospatial indexing system, a hexagonal hierarchical spatial index, has made significant strides in the field of discrete global grid systems. Initially developed to enhance Uber's ride-hailing services, such as dynamic pricing, H3 employs a hexagonal grid on an icosahedron's face, projected onto the Earth [31]. This structure simplifies analytical processes by utilizing hexagons for consistent neighbor distances. The system supports multiple resolutions, starting with 122 base cells and recursively subdividing for finer precision [32]. H3's flexibility and open-source nature have facilitated its application in diverse fields such as urban analysis, gaming, and augmented reality applications, including Pokemon Go [1]. Despite its advantages in operational flexibility and data integration, challenges with spatial accuracy and precision remain due to its hierarchical hexagonal structure and the gnomonic projection employed [1, 31, 32].

Modern geospatial data processing tools like the S2 Geometry Library and GeoHash are substitutes for more conventional approaches like Uber H3 and quadtrees [15]. These methods offer high pruning power, adaptable index data size, and support for effective updates of geographical data [15]. For instance, the S2 Geometry Library represents the Earth's surface using a hierarchy of cell covers, and GeoHash quickly retrieves position information by encoding geographic coordinates into a short string of characters and numbers [15]. Other research has also investigated the use of GeoHash indexes for spatial data models in corporate GIS [29] and suggested innovative methods for indexing and retrieving spatial polygons, such as the distributed KD-Tree [37].

S2, Uber H3, quadtrees, and other more modern spatial indexing algorithms are examples of advanced methods that the HierGP approach improves upon. In contrast to conventional techniques, which often use a non-adaptive grid system or a fixed resolution, HierGP uses an adaptive grid scaling strategy that dynamically adapts to the distribution and density of data. Because of its flexibility, HierGP can effectively handle large-scale geospatial datasets' volume variability and geographic complexity. Moreover, neither the S2 Geometry Library nor GeoHash fully addresses multi-dimensional support or multi-resolution analysis; these are features that HierGP incorporates. Through these innovations, HierGP addresses the nuanced demands of modern geospatial data analytics, offering scalability, precision, and operational flexibility improvements.

2.3 Differences of HierGP from Traditional quadtree Structures

Although HierGP shares some similarities with traditional quadtree indexing, several significant differences set it apart, particularly in adaptability, data processing, and support for large and complicated datasets.

Adaptive Grid Sizing. HierGP uses adaptive grid sizing instead of the uniform subdivision technique found in typical quadtrees, which divide each node into four equal-sized quadrants. By using a recursive partitioning technique similar

to that of quadtrees, the grid can dynamically change according to the distribution and density of data in each location. This flexibility ensures finer partitions for areas with higher data density, thereby enhancing the accuracy and efficiency of the spatial index.

Data-Driven Partitioning. HierGP incorporates data features like variance and clustering to improve partitioning decisions. In contrast to quadtrees, which use only geographical criteria, HierGP uses data-driven criteria to identify better underlying data patterns. This feature benefits applications where data are naturally heterogeneous, including urban planning and environmental monitoring.

Handling of Large Datasets. Large datasets can make traditional quadtrees inefficient, leading to deep tree topologies that complicate traversal and querying. HierGP incorporates methods for managing massive volumes of data, enhancing scalability.

Multi-Resolution and Multi-Dimensional Support. Another critical improvement over conventional quadtrees is that HierGP can handle multi-dimensional data and support multi-resolution queries. Due to its adaptability, HierGP can be used for various tasks, such as analyzing time-series data in environmental studies and 3D geographic information systems (GIS), where multiple resolutions and dimensions are vital for analysis.

3 MAP POINT REDUCTION

The objective of our proposed *MPR* algorithm is to quickly and accurately downsize the number of unique *CRS* values within an extensive dataset so that groupings of similar observations within a user-defined grid size distance of one another can be represented by a single *CRS* point.

Developing and implementing an effective *MPR* algorithm presents several technical challenges. The primary challenges focused on in this paper include:

(1) Efficient mapping of *CRS* values to a model of **orthodromic distance** (the shortest surface distance between two points themselves on the surface of a sphere) for Earth.
(2) Computational time complexity of the *MPR* process execution, including parallelizable mapping capacity.
(3) Ability to mini-batch samples from an exceedingly large and continuously expanding dataset into a single consistent *MPR* result.
(4) Determining the appropriate grid sizing for the dataset to maintain the desired accuracy criteria across *CRS* point aggregations.

3.1 Simplified Model of Earth

To address the problem of accurate mapping of *CRS* values to Earth's orthodromic distance, we start with the creation of a simplified model of Earth. This model considers two orthogonal axes made of circumference (circ.) values, one for latitude and the other for longitude:

(1) **Equatorial** circ. of Earth as $C_e = 40,075.017$ km.
(2) **Polar** circ. of Earth as $C_p = 40,007.863$ km.

The two circumference values are both considered as uniform perfect circles. This assumption maintains simplicity in the model while allowing a better approximation of Earth's spherical shape.

Using this representation of Earth, the relationships between degree changes from the Earth's center to distance changes along each C_e and C_p can be easily computed. Trivially, each C is the surface distance a 360-degree shift yields

along each model's axes. From this, the distance that a single degree shift (Δ) along each axe would yield can be found using the following formulation: $\Delta = \frac{C \text{ km}}{360 \text{ degrees}}$. Using this formulation Δ_e for Equatorial and Δ_p for Polar can be found as:

$$\Delta_e = \frac{40,075.017 \text{ km}}{360 \text{ degrees}} = 111.3194917 \text{ km / degree} \tag{1}$$

$$\Delta_p = \frac{40,007.863 \text{ km}}{360 \text{ degrees}} = 111.1329528 \text{ km / degree} \tag{2}$$

To calculate the required degree shift along each axis to achieve a specific grid size (denoted as s), we define a term called delta (δ). This δ represents the orthodromic distance in degrees needed per grid unit. It is determined using the following formula:

$$\delta = \frac{s \text{ km}}{\Delta \text{ km/degree}} \tag{3}$$

This formula calculates the degree shift (δ) required along each axis to match the grid size s based on the degree distance Δ per kilometer.

3.2 The MPR Algorithm

The computed δ values are designed to be computed only once by the algorithm and then stored in a configuration instance for repeated use, allowing the algorithm to benefit from parallelization. It is also important to note that the calculated deltas may result in minimal values where the significant digits are farther from the decimal point. Because of this, users can adopt a simple approach to better maintain accuracy in mappings for the *MPR* process. Such a measure allows users to specify a parameter *acc*, indicating how many decimal places of accuracy they want for the computed δ values. The *MPR* process then can truncate or round (depending on implementation details) the computed δ values to *acc* decimal places. Using the above formulations of s to δ relation, a new resulting s as s' can be solved using these new δ values. Then s' can be recommended to the user to use directly or revise. By managing the algorithm's delta values and implementation details in such ways, users can better mitigate computational errors, including truncation and round-off.

While the resulting grid size will likely not be square ($s'_e \neq s'_p$), users could account for the resulting rectangular grid size to reduce the cumulative arithmetic error, which is most impactful on relatively small grid size mapping requests. Either way, the capacity for rectangular grid sizing is a built-in offering of the algorithm that provides much flexibility.

Starting from a *CRS* point valued ($latitude = 0.0, longitude = 0.0$), the stored δ' values can function as step sizes along each axis in all four directions. Consequently, an implicit grid system is formed that spans the entire *CRS* value bounds (e.g., longitude $\in [-180, 180]$, latitude $\in [-90, 90]$). This way, the entire space for all possible *CRS* points is partitioned into a two-dimensional grid of cells. Each cell in the grid is implicitly assigned a unique indexing (X, Y) following a zero-based indexing scheme, where the first positive cell in X and Y is $(0, 0)$. The left pane in Figure 1 shows the cell indexing scheme of the resulting implicit grid system.

With a grid indexing system now in place, the set of *CRS* points in a dataset, when considered as two vectors ($\vec{lat}$ for latitude and $\vec{lon}$ for longitude), can be assigned grid cell indices in a vectorized fashion via the following formulation:

$$\vec{X} = \left\lfloor \frac{\vec{lon}}{\delta_e} \right\rfloor, \vec{Y} = \left\lfloor \frac{\vec{lat}}{\delta_p} \right\rfloor \tag{4}$$

Here, $(\vec{X}[i], \vec{Y}[i])$ is the assigned grid cell index for the *CRS* observation at index i with coordinates $(\vec{lon}[i], \vec{lat}[i])$.

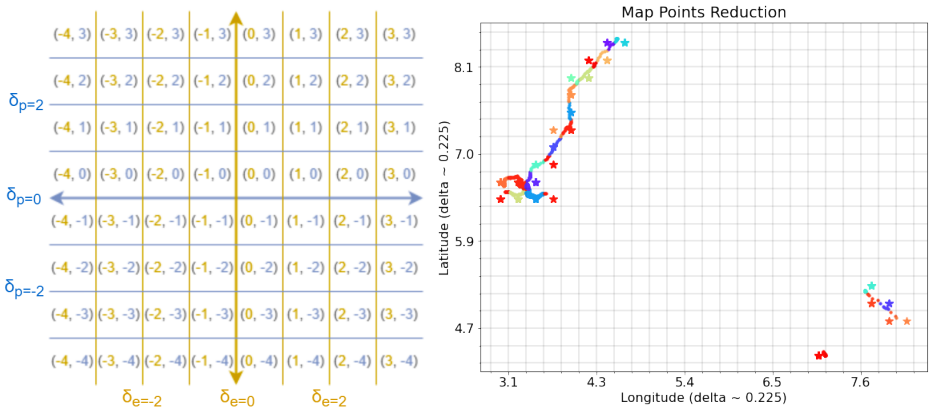


Fig. 1. **Left**: Cell indexing scheme of the implicit grid system. **Right**: Example result as a Cartesian graph for a dataset with 5,000 initial unique CRS values (dots) reduced to 26 unique CRS values (stars) using a grid size s=25 km and cell centers for midpoints.

Grid cell indices can then be easily converted into *CRS* points relative to their assignments in a vectorized manner. While a user could use any point relative to grid indexing, one obvious choice for representation is simply the midpoint *CRS* values of each cell. The midpoint assignments, denoted as $(\vec{mid}_{lat}, \vec{mid}_{lon})$ can be found via:

$$\vec{mid}_{lat} = \delta_{lat} \cdot (\vec{Y} + 0.5), \vec{mid}_{lon} = \delta_{lon} \cdot (\vec{X} + 0.5) \tag{5}$$

Given that points could be clustered together on one side of a cell, a better representative choice for a midpoint selection can be the computed *centroid* of the *lat* and *long* values of all points within the cell.

Now $(\vec{mid}_{lat}[i], \vec{mid}_{lon}[i])$ for the *CRS* point at index i are the new *CRS* values assigned by the *MPR* algorithm to represent that point. The right pane in Figure 1 shows an example *MPR* result as a Cartesian graph for a dataset with $5,000$ initial unique *CRS* input values (dots) reduced to 26 unique *CRS* output values (stars) using a grid size of 25 *km*. This example dataset had a range of $[4.3835007, 8.4808539]$ for latitude and $[3.0718008, 8.0892264]$ for longitude. Algorithm 1 summarizes the computational steps of the approach. Note that it does not include any adjustments for the potential computational errors incurred in delta calculations as previously outlined.

Algorithm 1 Map Point Reduction Algorithm (w/o delta adjustment)

Input: D (n x 2 matrix of CRS points, n = number of samples), s (grid size in km where $0.0 < s \in \mathbb{Q}$)
Output: D' (n x 4 matrix of initial and new CRS points)

1: Establish $\Delta_e = 111.3194917, \Delta_p = 111.1329528$
2: Compute $\delta_e = \frac{s}{\Delta_e}, \delta_p = \frac{s}{\Delta_p}$
3: Extract latitude values from D into $\vec{lat}$
4: Extract longitude values from D into $\vec{lon}$
5: Compute $\vec{x} = \left\lfloor \frac{\vec{lon}}{\delta_e} \right\rfloor, \vec{y} = \left\lfloor \frac{\vec{lat}}{\delta_p} \right\rfloor$
6: Compute $\vec{mid}_{lat} = \delta_p \cdot (\vec{y} + 0.5), \vec{mid}_{lon} = \delta_e \cdot (\vec{x} + 0.5)$
7: $D' \leftarrow$ Concatenate $D, \vec{mid}_{lat}, \vec{mid}_{lon}$
8: **return** D'

4 HIERARCHICAL REPRESENTATION

The proposed approach is efficient; however, processing the entire geospatial dataset at once may not be feasible due to its size. To address this, we suggest a hierarchical representation of spatial and temporal partitioning using a mini-batching approach. This means the large dataset can be divided into smaller non-overlapping subsets, allowing for iterative processing until the entire set is covered. To ensure the validity of any reduction, the samples must represent the entire dataset across all its features. By assigning the dataset to a single hierarchical representation, reduction tests can be conducted locally within partitioned groups. Our HierGP approach allows for the development of semantically interpretable heuristics, which can be applied to future collected samples. Additionally, researchers can periodically repeat this process to keep the learned heuristics current and relevant to new samples.

4.1 Spatial Partitioning

The initial *CRS* points are considered layer zero (l_0), serving as a baseline for later analysis. By passing all of l_0's *CRS* points into the *MPR* method with an initial grid size s_i (e.g., $100m$), a new set of *CRS* points, layer one (l_1), is generated. Next, by following a quadtree-based approach, additional tree layers can be built iteratively, layer by layer. Each higher layer l_i can be built by using the *CRS* points of the previous layer l_{i-1} and passing them into the *MPR* method using a value for s_l that is an integer factor $\alpha > 1$ larger than s_{l-1} (the previous layer's value for s).

More formally, layer l_i is added atop a layer l_{i-1} with grid size s_{i-1} by performing the following steps:

(1) $s_l = \alpha s_{l-1}$ (e.g. if $\alpha = 2$ and $s_i = 100m$, then $s_1 = 200m$, $s_2 = 400m$, $s_3 = 800m$, ...)
(2) Pass the grid assignment *CRS* values (e.g. $\vec{mid\_lat}$ and $\vec{mid\_lon}$) of l_{i-1} into the *MPR* algorithm.
(3) Assign the *CRS* values output from *MPR* to layer l_i.

It is recommended to keep adding layers until the desired tree depth is reached or all *CRS* points are assigned to a single grid cell. It is suggested to use a static value for grid size throughout the tree analysis; for example, doubling the grid size with each layer added. Figure 2 illustrates a simple example of the hierarchical tree structure of grid cell partitions using a static parent-child grid size relationship of doubling the size in each step, following the classic quadtree approach.

4.2 Temporal Partitioning

Environmental datasets are typically collected at certain rates, which can be regular (hourly, daily, yearly) or irregular. It's important to examine the data collection rate to determine how to partition the data over time. Temporal partitioning reduces data along a dimension orthogonal to spatial grid partitioning (e.g., latitude and longitude). It allows for simplifying data points in a grid that exist at different times using a user-selected temporal resolution (τ) that matches the dataset's temporal resolution. The number of temporal partitions, which we denoted by k, depends on the range of *datetime* values in the dataset and the chosen τ. To find k, we calculate $k = \lceil \frac{\tau_{max} - \tau_{min}}{\tau} \rceil$, where τ_{min} is the earliest *datetime* value and τ_{max} is the latest. Figure 3 shows an example of a spatial grid cell partition with five temporal partitions ($k = 5$ for some τ). Algorithm 2 summarizes the vectorized operations for computing the temporal partition assignments.

4.3 Hierarchy Level Heuristics

A hierarchical representation is used to store the spatially and temporally partitioned dataset in a series of layers across different features. This allows for quick consideration of different groups of points for holistic similarity across the entire feature space, rather than focusing solely on spatial or temporal aspects.

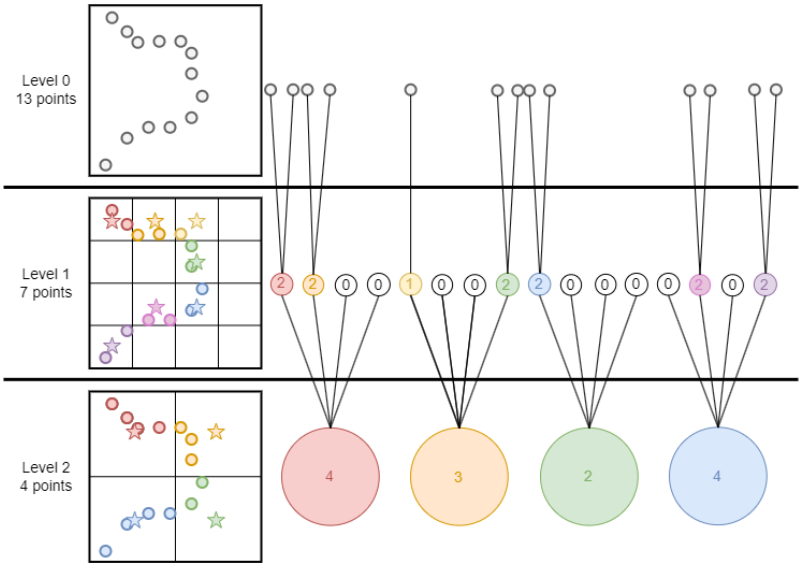


Fig. 2. Illustrated Example of the Hierarchical Tree Structure of Grid Cell Partitions.

Algorithm 2 Temporal Partitioning

Input: Temporal dataset D (n x 1 matrix of *datetime* values, n = number of samples), Temporal resolution τ
Output: Temporal partition assignments T (n x 1 matrix)

1: $T = \left\lfloor \frac{D - \tau_{min}}{\tau} \right\rfloor$
2: **return** T

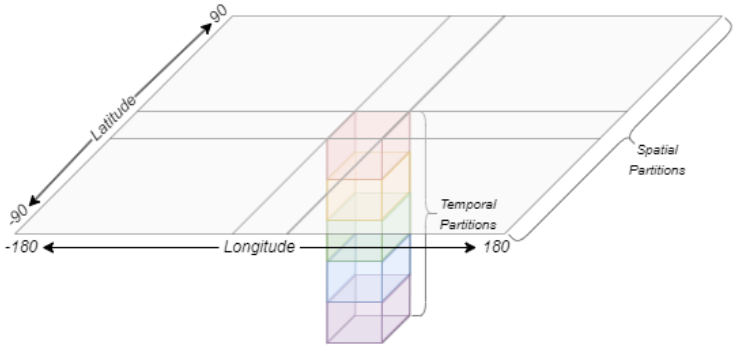


Fig. 3. Illustration showing the relationship between temporal and spatial partitioning.

Starting from the highest layer in the hierarchy, many or all points in the dataset fall into a single partition. As we move down the hierarchy levels, each partition of points is recursively subdivided into finer spaces in a nested fashion until each point in the dataset is alone in its own partition.

When each partition in the hierarchy is visited, a representative point can be created from the subset of points within the partition space. This reduced point can be evaluated against a user-defined measurement accuracy condition (ϵ). If

Algorithm 3 Hierarchical Grid Partitioning Algorithm

Input: Original dataset D_o (n x 3 matrix of spatial-temporal samples, n = number of samples), ϵ (measurement accuracy condition), s_i (initial grid size), s_m (maximum grid size), α (integer for grid size level scaling), τ (temporal resolution)
Output: Reduced dataset D_r

Compute list of grid sizes $\vec{g} = [s_i, \alpha s_i, \alpha^2 s_i, \ldots, s_m]$
Perform temporal partitioning (Algorithm 2) on D_o using τ into a list of matrices $d = [D_o^{t=0}, D_o^{t=1}, \ldots, D_o^{t=k}]$
Initialize empty matrix D_r for all reduced points
for each D_o^i in d **do**
    Set unreduced dataset matrix $D_u \leftarrow D_o^i$
    Set empty reduced dataset matrix D_r^i
    for each grid size level s_l in reverse order of $\vec{g}$ (s_m to s_i) **do**
        Perform *MPR* (Algorithm 1) of all data points in D_u using s_l
        for each partition p in level l **do**
            Reduce all points K in p into a single point r_p^l
            if r_p^l satisfies ϵ **then**
                Add r_p^l to D_r^i
                Remove K from D_u
            end if
        end for
        if all points are successfully reduced (D_u is empty) **then**
            Add D_r^i to D_r
        end if
    end for
    if unreduced points remain (D_u is not empty) **then**
        Add D_u to D_r^i
    end if
    Add D_r^i to D_r
end for
return D_r

the condition is satisfied, then that spatial-temporal space can be reduced; otherwise, the points cannot be reduced at that level.

Using this framework, specific loss relative to ϵ can be tracked for each partition, and an analysis of the results can be used to develop heuristics from the underlying dataset. Heuristics based on past data patterns and predefined accuracy levels guide the grid size adjustments to provide a dynamic partitioning architecture that enables customized algorithmic responses to varying data features.

This procedure improves customization and adaptation to specific user requirements and data situations. A formal and detailed description of the procedure is provided in pseudo-code format in Algorithm 3.

Figure 4 illustrates how the *MPR* hierarchy can be used to develop a variable (or uniform) grid size heuristic based on a set of customized accuracy threshold conditions. Figure 5 shows visualizations of two synthetically created datasets of varying sizes (kept small for plotting purposes) and how the method can be applied to recommend reductions up to the specified level using two different criteria.

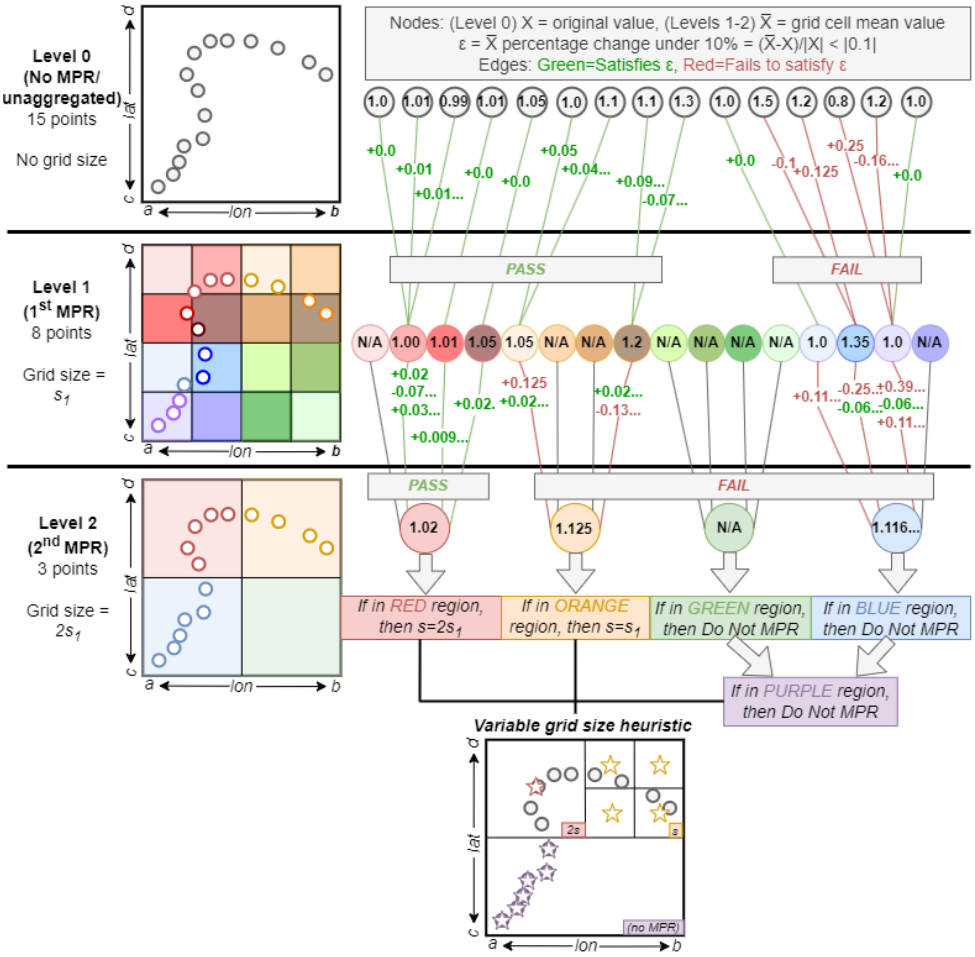


Fig. 4. Illustration showing the use of *MPR* hierarchy for developing variable grid size heuristics based on the use of a customized measurement accuracy threshold ϵ for analysis of one feature of the data X.

5 COMPARISON OF HIERGP WITH OTHER GEOSPATIAL ALGORITHMS

We compared HierGP with several leading geospatial data processing algorithms, including Uber H3, S2 Geometry Library, and GeoHash. The aim was to evaluate their performance across three critical aspects: execution time, memory usage, and scalability across grid levels. This section details the methodology of the comparison, discusses the results obtained, and provides insight into the efficiency of each algorithm under various computational scenarios.

5.1 Methodology

Satellite image data interpretation. When interpreting data from images where pixels represent regions of land, the geometric configuration of the processing grid can affect the efficacy of data interpretation. Since pixels are an atomic unit, their dimensions are easily interpreted as a grid of squares. Depending on the level of magnification, the pixels can each represent an X by X region of land. When employing a hexagonal mesh overlaid on a matrix of pixel values,

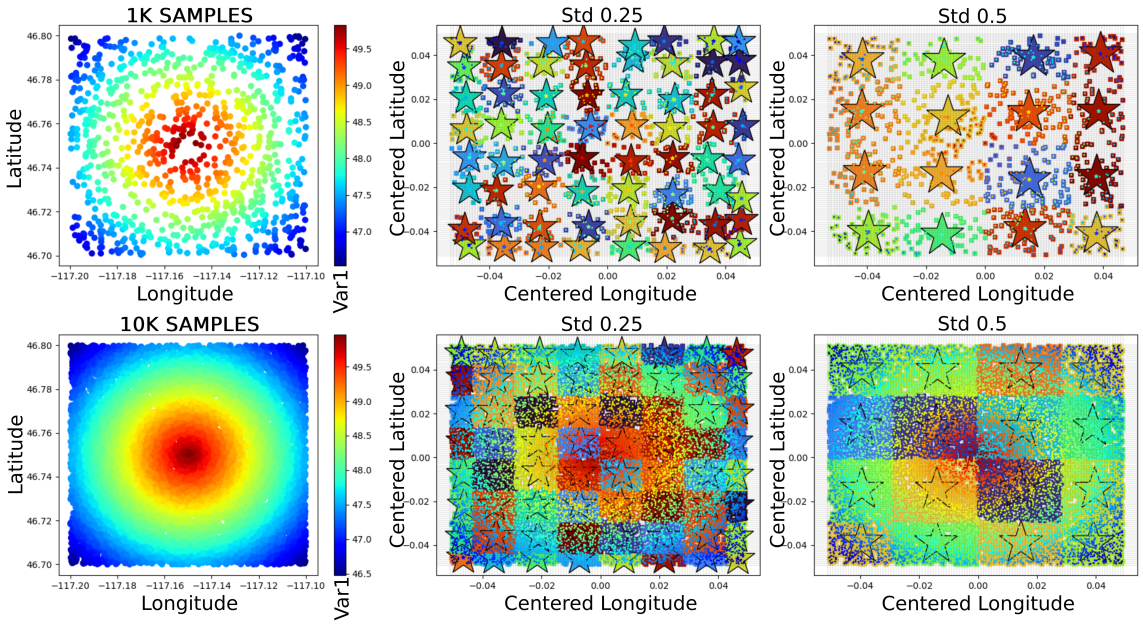


Fig. 5. **Left:** Heat maps of 1k (top) or 10k (bottom) synthetically created geospatial data samples over an area approximately 7.6 by 11 km and a single variable 'var1' [45-50]. **Center:** The same points are reduced to grids using a HeirGP instance with the smallest grid size of 0.1km (level 1) and the largest grid size of 12.8km (level 7), where each level doubles the previous. Each point's square exterior color reflects its grid cell assignment, while its interior dot is its 'var1' value. The stars represent the reduced points with the exterior color reflecting grid cell association and the interior the average 'var1' value when all points of the same exterior color are reduced and become it. The metric for reduction eligibility is if the standard deviation (std) of the samples in a grid cell is ≤ 0.25, then reduce. **Right:** The same as the center but with more relaxed reduction criteria of std ≤ 0.5.

the translation is anything but trivial. There are generally three considerations when overlaying the hexagonal grid over the square one:

(1) **Size**. The side length relation can be found via the following steps. Let a be the length of a side in the hexagon. The area of a hexagon is then given by $\frac{3\sqrt{3}}{2}a^2$. Let b be the length of a side in the square. The area of a square is then trivially b^2. If both areas are set equal to each other for the best possible coverage and we solve for b, we get $b = \frac{\sqrt[4]{3^3} \cdot a}{\sqrt{2}}$ since side length would be positive. For example, if a were 1km, b would be approximately 1.6km.
(2) **Position**. Using a chosen midpoint, offset the hexagonal grid so that each cell's area aligns with the rows and columns of the square grid.
(3) **Rotation**. Rotating the hexagonal grid overlay by 45 degrees could marginally improve the fit.

Whatever the approach, moving from pixels to square grids is met easily via scaling to match the dataset's metadata (e.g., each pixel is 30x30km). However, matching hexagonal mesh to such an image is more difficult, and no matter the approach taken, it results in a mismatched representation. Figure 6 illustrates how the two grid systems could be applied to an image file and the resulting differences in outcome.

Comparison. We assessed execution times using datasets ranging from 1,000 to 10,000,000 records to gauge the performance of each algorithm as the data volume scaled up. This comparison was pivotal to understanding each system's responsiveness to increasing data sizes. Scalability was tested by adjusting the grid level parameter from 1 to

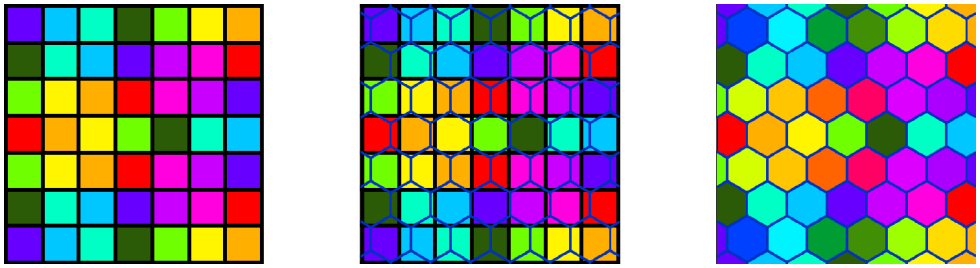


Fig. 6. Illustrates showing square and hexagonal mesh grid interpretations of a discrete pixel-value matrix. **Left** is square grid interpretation. **Center** is a possible overlay placement approach. **Right** is a possible resulting interpretation that adopts a color value of the average of colors underneath each hex grid.

15 while keeping the dataset size constant (1 million records). This metric showed how each algorithm's performance scales with increased complexity. Finally, we monitored the memory usage for each algorithm across different dataset sizes using memory profiler and psutil libraries in Python. This measure was crucial for assessing each algorithm's resource utilization efficiency. Memory usage was measured by calculating the difference in peak memory before and after the algorithm's execution, focusing on the memory allocated for storing the data structures.

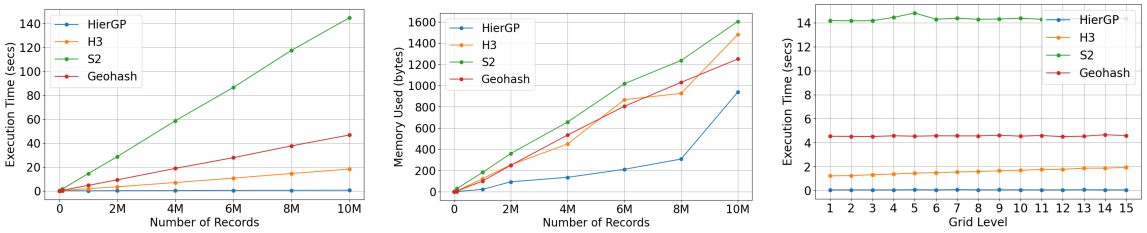
5.2 Results and Analysis

Execution Time. The execution time results are presented in Figure 7a. HierGP consistently outperformed H3, S2 Geometry Library, and GeoHash in execution time across all dataset sizes, with its advantage growing as the number of records increased. This suggests that HierGP's performance benefits significantly from its data structure alignment with typical image pixel layouts.

Memory Usage. The memory usage results, depicted in Figure 7b, indicate that HierGP is more memory-efficient than the other algorithms. This efficiency is particularly pronounced at higher record counts, where HierGP's memory footprint remains modest compared to H3's significantly larger usage. The memory measurement reflects the size of the data structures generated by each algorithm and their overall storage efficiency.

Grid Level Scalability. As shown in Figure 7c, all algorithms demonstrated stable execution times across varying grid levels while maintaining a constant dataset size of 1 million records. HierGP consistently maintained a lower execution time throughout these tests. These results highlight the importance of data structure compatibility with the data being processed, where HierGP's square grid design aligns more naturally with satellite data pixels.

We note that the hexagonal shape of H3 cells introduces complexity in data alignment and can increase computational overhead, particularly when converting from square grids used in typical satellite images. We did not explicitly consider any initial conversions from square to hexagonal grids in our execution time results. Such modifications may be a contributing factor to the observed variations in memory consumption and execution times if they take place within the algorithms. This suggests that the underlying processing techniques of the algorithms may directly impact performance measures.



(a) Scalability: Execution Time (b) Scalability: Memory Usage (c) Scalability: Grid Level

Fig. 7. Scalability comparisons between HierGP and other algorithms across various performance metrics. **(a)** execution time, **(b)** memory usage, and **(c)** grid level scalability

5.3 Discussion

The comparative analysis demonstrates that HierGP has clear advantages over the other algorithms in terms of execution time, memory footprint, and scalability for large-scale geospatial data processing tasks. The differences in grid shape—HierGP's square grid versus H3's hexagonal grid—play a crucial role, particularly in environmental satellite data. Additionally, the parallel nature of the HierGP algorithm makes it amenable for parallel and distributed computing scenarios.

Moreover, HierGP's square-based design is inherently compatible with the pixel structure of satellite data, providing a more natural fit for this type of data. The square grid aligns closely with the pixel layout, potentially leading to more intuitive data integration and analysis.

The implementation languages of these algorithms also differ. The H3 algorithm was initially developed in C++ and then converted to a Python module. Python was used to directly generate HierGP and implement GeoHash and the S2 Geometry Library. These linguistic variances significantly impact the ease with which the algorithms can be incorporated into workflows for data processing. In Python-centric setups, Python algorithms such as HierGP, S2, and GeoHash might make development more accessible. Conversely, algorithms like H3 with parts initially written in C++ can gain from the superior performance of C/C++ via libraries like numpy. The varied underlying languages used by the algorithms indicate varying degrees of integration ease and possible performance.

6 DEMONSTRATIONS ON APPLICATIONS

This section demonstrates the benefits of the HierGP framework using two real-world applications. The first application involves environmental health analysis, while the second deals with environmental data extraction for large polygons.

6.1 Environmental Health Analysis

We begin by briefly describing the dataset and then present the result of the partitioning process. We first consider the partition in terms of the number of grids generated at each level and explain why the counts differ. Then, we consider the computational time and space complexity of our approach and finally combine the spatial and temporal partitioning.

6.1.1 Brief Summary of the Dataset. The dataset is an extensive collection of location data from participants in a twin registry. A recent study explored the possibility of using the Google Location History (*GLH*) of participants to obtain location data [10]. Participants can provide their *GLH* by exporting information from their Google account. This data contains their raw location data and semantic location data, which Google has augmented with the most probable

Table 1. Grid counts across tree levels for the total 289,446,156 points.

Tree Level	Grid Size	Count
1	$50m$	7,380,968
2	$100m$	4,383,522
3	$200m$	2,541,608
4	$400m$	1,417,877
5	$800m$	742,318
6	$1.6km$	377,356
7	$3.2km$	192,162
8	$6.4km$	95,755
9	$12.8km$	45,709
10	$25.6km$	20,616
11	$51.2km$	8,693
12	$102.4km$	3,687
13	$204.8km$	1,622
14	$409.6km$	745

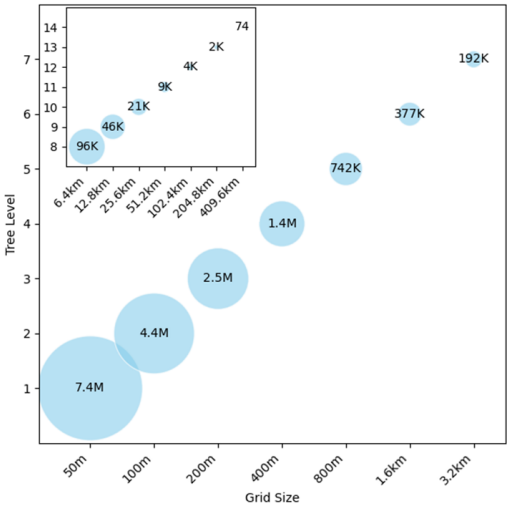


Fig. 8. Bubble chart visualizing the point counts across tree levels shown in Table 1

activities and places they visit. In this paper, we focus on the raw location data. The dataset comprises 372 participants and 289,446,156 location points from May 2010 to December 2022.

6.1.2 Spatial and Temporal Partitioning Results. Using the *MPR* algorithm, all 289 million *CRS* points were partitioned into a grid hierarchy of 14 levels, starting with a grid size of $50m$. Each level increment doubles the grid size from the previous level. The highest level in the hierarchy (level 14) has a size of $409.6km$. At the first level of partitioning, we obtain a 97.45% reduction. Table1 shows the number of grids at each level and Figure 8 visualizes the same data using a bubble chart. In addition to the spatial partitioning, we also did temporal partitioning using yearly and hourly temporal resolutions. Table 1 and Figure 8 only show spatial partitioning results.

6.1.3 Resolution Combination Results. We examined two target datasets from Google Earth Engine: the ERA5-Land Hourly - ECMWF Climate Reanalysis (*ERA5-Land*) [7] and the MOD13Q1.061 Terra Vegetation Indices 16-Day Global 250m (*MOD13Q1*) [8].

The *ERA5-Land* dataset contains several land variables, including temperature, wind speed, precipitation and surface atmospheric pressure. This dataset has a spatial resolution of $11{,}132m$ and an hourly temporal resolution. A general rule of thumb for selecting the best grid without using hierarchy level heuristics is to choose a grid size that is less than half the spatial resolution of the target dataset. An appropriate grid level for this dataset would be $3.2km$.

The *MOD13Q1* dataset contains vegetation indices on a per-pixel basis. The dataset variables include the Normalized Difference Vegetation Index (NDVI), a popular metric for quantifying the health and density of vegetation. This metric uses sensor data for quantifying the difference between near-infrared (reflected by vegetation) and red light (absorbed by vegetation), normalized from -1 (likely water) to 1 (likely dense green vegetation) [6]. The dataset has a spatial resolution of $250m$ and an annual temporal resolution. Without the use of any heuristics, an appropriate grid level would be $100m$. Next, we determine the best grid level using the hierarchy heuristics process outlined in Section 4.3.

Table 2. Spatial and Temporal Resolution Combinations

Identifier	Spatial	Temporal	Count	% of Data (289 million)
l2-y	$100m$	yearly	13,849,364	4.78%
l7-h	$3.2km$	hourly	22,461,118	7.76%

Table 3. Means and standard deviations of the percentage differences for all the sample points for NDVI and temperature variables.

		NDVI		**temperature**	
Level	km	$\bar{X}$ (%)	σ_x (%)	$\bar{X}$ (%)	σ_x (%)
1	0.05	**-0.17**	**4.53**	**0.00**	**0.01**
2	0.1	**-0.27**	**6.26**	**-0.02**	**1.41**
3	0.2	**-0.86**	10.34	**-0.02**	**1.41**
4	0.4	**-0.73**	12.5	**-0.01**	**1.00**
5	0.8	**0.00**	18.24	**-0.14**	**3.74**
6	1.6	2.86	27.59	**-0.90**	**9.45**
7	3.2	3.14	38.26	**-0.79**	**8.85**
8	6.4	**-0.48**	68.16	**-1.52**	12.24
9	12.8	6.19	136.96	**-1.78**	13.22
10	25.6	8.32	140.10	**-4.81**	21.38
11	51.2	29.73	151.24	-13.84	30.75
12	102.4	33.78	160.59	**-3.58**	18.01
13	204.8	26.16	131.64	**-3.24**	17.00
14	409.6	-40.99	173.90	**-1.23**	**9.61**

Table 2 shows the spatial and temporal combinations for the selected levels for *ERA5-Land* and *MOD13Q1*. After combining the spatial and temporal combination, the count column shows the resulting number of records, while the percentage column shows the percentage of the original 289 million records. Essentially, this means that instead of having to run 289 million matches against the target dataset, this method can reduce it to less than 5% of the data for the *MOD13Q1* and less than 8% for the *ERA5-Land*.

6.1.4 Hierarchy Heuristics Results. To determine the best hierarchy level using the heuristics approach defined in Section 4.3, we sampled 10,000 records from the *CRS* points. While this might be considered relatively small compared to the total number of records in the dataset, it was sufficient to determine which hierarchy level is best for the data. We set the cut-off accuracy condition ϵ to the following: the mean of the value shift should not exceed a range of ±2%, the value shift standard deviation should not exceed 10%, and all previous levels must also satisfy this condition. With the accuracy condition defined, we extracted NDVI values from the *MOD13Q1* and temperature values from the *ERA5-Land* for the 10,000 points.

Next, we reduced the points using the *MPR* process into the first 50m grid and extracted values for the grid centers of those points. We continued this process for all the 14 levels of the grid. Once all the values were extracted successfully, we calculated the mean and standard deviation of the value shift, comparing them with those extracted when directly using the *CRS* points. Table 3 shows the means and standard deviations of the value shift for the NVDI and temperature values extracted.

Table 3 also highlights the levels that match the specified accuracy condition in bold. For example, the mean and value shift for levels 1 through 5 and 8 satisfy the accuracy condition, while levels 1 and 2 only meet the standard deviation and value shift. All levels except level 11 satisfy the mean accuracy condition for the temperature extract, while levels 1 through 7 and 14 satisfy the standard deviation accuracy condition.

We check the highest level that satisfies the accuracy condition to determine the best level. For the NDVI extraction, the highest level that satisfies the condition is level 2, i.e., the $100m$ grid size (this is marked in blue in Table 3). Similarly, for the temperature extraction, the highest level that satisfies the condition is level 7, i.e., the $3.2km$ grid size (marked in green in table 3). The resulting heuristic recommends using grid size $s = 0.1\ km$ for NDVI analysis and $s = 3.2\ km$ for temperature analysis on the larger dataset.

6.2 Environmental Data Extraction for Large Polygons

The extraction of NDVI values from Landsat imagery, particularly over large areas like counties in the United States, presents significant computational challenges. NDVI, a critical indicator of vegetation health, is crucial in agriculture and forestry. The traditionally resource-intensive process encounters difficulties in large-scale computation due to cloud cover, topographical variations, and data sparsity. Therefore, we have adopted a grid-based subdivision approach using the HierGP algorithm. This method optimizes processing by breaking down large areas into smaller, manageable sections, enhancing data accuracy at the borders and contributing a novel solution to the computational difficulties in extensive NDVI analysis.

6.2.1 Data Extraction Process. The NDVI extraction process from Landsat imagery entails a detailed sequence of steps. Initially, it involves selecting and extracting relevant Landsat images for the specified region and period. The critical aspect of computing NDVI is analyzing of the Landsat image bands, specifically, calculating the normalized difference between the near-infrared (B4) and red light (B3) bands. This calculation is essential as it transforms raw satellite data into meaningful metrics that effectively indicate the health and vitality of vegetation in the targeted area.

6.2.2 HierGP Application. The process of NDVI extraction from Landsat images is significantly enhanced when dealing with large areas using the HierGP algorithm. This method begins by segmenting the expansive polygon into smaller, manageable grids. The initial step involves classifying the polygon points into respective grids, focusing primarily on the boundary areas. This classification forms the basis for filling the internal grids, using the border grids as a reference. Particular attention is given to these border grids, which require clipping to align precisely with the polygon's shape, as depicted in Figure 9. Each grid, once defined, is processed individually, allowing for the extraction of NDVI values in a manner that is both efficient and tailored to the specific shape and size of the polygon. The final step in this optimized approach involves aggregating the results from each grid to obtain a comprehensive NDVI analysis for the entire polygon. This approach effectively addresses the computational challenges and potential time-outs associated with processing large geographical areas, such as those encountered on platforms like Google Earth Engine.

6.2.3 Other Optimization Strategies. In addition to employing HierGP, several optimization strategies were implemented. A combined reducer approach minimized the number of Earth Engine invocations. Conditional masking was applied only when necessary, and clipping was performed at the image collection level to leverage Earth Engine's capabilities, reducing processing times.

6.2.4 Extraction Results. Implementing the HierGP framework markedly improved the NDVI extraction process. This improvement generated a comprehensive NDVI summary at the polygon level, offering in-depth insight into vegetation

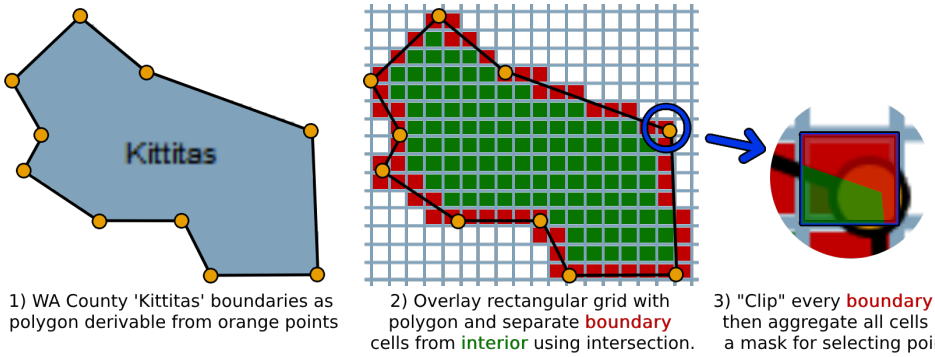


Fig. 9. Illustration of general extraction process of points using HierGP over basic geographical polygon of Kittitas County is WA.

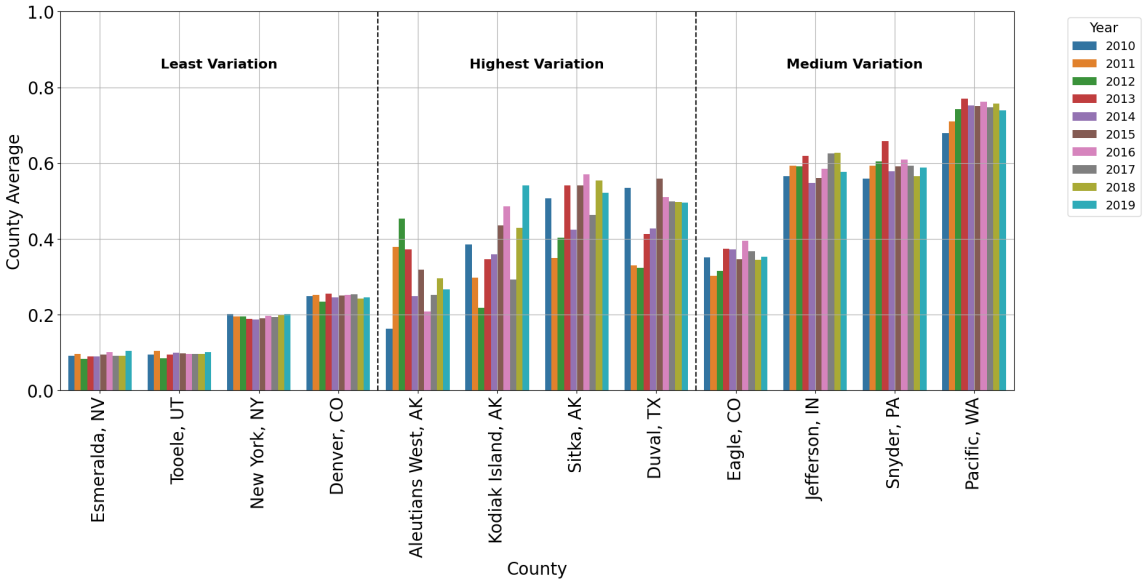


Fig. 10. Distribution of County Average NDVI Values by Year for Counties with Least, Highest, and Medium Variations. This bar plot groups 12 counties into three categories: Least Variation (first four counties), Highest Variation (middle four counties), and Medium Variation (last four counties). Each bar represents the average NDVI value for a specific county and year.

health across extensive regions. The effectiveness of HierGP is further highlighted by its ability to process large datasets efficiently, significantly reducing execution time and memory usage. Figure 10 provides a comparative analysis illustrating the average NDVI values in twelve carefully selected U.S. counties. These counties were chosen to cover a broad range of data variability, from small variations to high variations in values. It would not have been possible to generate summary statistics and perform exploratory data analysis for such a large dataset without employing the HierGP approach.

7 CONCLUSION

We introduce a novel dynamic global grid system called Hierarchical Grid Partitioning, HierGP, that offers a flexible and fast approach for creating a customized global grid system. Unlike traditional discrete global grid systems, HierGP enables users to customize grid sizes and structures to suit specific analytic needs. In our extensive case study, HierGP enabled efficient processing of over 289 million spatiotemporal data points from phone location data for more than 350 individuals. We also demonstrated HierGP's use in extracting environmental data for large polygons, specifically the extraction of NDVI values for United States counties. This reinforces the framework's versatility and role in advancing geospatial analytics. Furthermore, HierGP was shown to have superior performance compared to state-of-the-art systems. HierGP stands out not only for its high speed and low memory footprint, but also for its ability to handle vast spatial-temporal data streams in near-real time. This capability is particularly beneficial for emerging environmental analyses, such as those needed for smart cities, climate resilience, and ecosystem health services that span large geographic and temporal domains.

The GitHub repository containing the HierGP algorithm developed in this study and the associated code for generating the visualizations is publicly available at https://github.com/funsooje/HierarchicalGridPartitioning.

ACKNOWLEDGMENTS

Research reported in this publication was supported by the National Science Foundation under Award Number 2126449 and by the National Institute Of Environmental Health Sciences of the National Institutes of Health under Award Number R21ES031226. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

REFERENCES

[1] Mark Altaweel. 2024. H3: OPEN SOURCE GEOSPATIAL INDEXING SYSTEM. Retrieved January 24, 2024 from https://www.geographyrealm.com/h3-open-source-geospatial-indexing-system/

[2] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. 2016. *Recent Advances in Graph Partitioning*. Springer International Publishing, Cham, 117–158. https://doi.org/10.1007/978-3-319-49487-6_4

[3] Chung-I Chang, Nancy P Lin, Nien-Yi Jan, et al. 2009. An axis-shifted grid-clustering algorithm. *Journal of Applied Science and Engineering* 12, 2 (2009), 183–192.

[4] Gabriel Felbermayr, Jasmin Gröschl, Mark Sanders, Vincent Schippers, and Thomas Steinwachs. 2022. The economic impact of weather anomalies. *World Development* 151 (2022), 105745. https://doi.org/10.1016/j.worlddev.2021.105745

[5] Puspendu Ghosh and Mala De. 2022. A comprehensive survey of distribution system resilience to extreme weather events: concept, assessment, and enhancement strategies. *International Journal of Ambient Energy* 43, 1 (2022), 6671–6693. https://doi.org/10.1080/01430750.2022.2037460 arXiv:https://doi.org/10.1080/01430750.2022.2037460

[6] GISGeography. 2024. What is NDVI (Normalized Difference Vegetation Index)? Retrieved April 20, 2024 from https://gisgeography.com/ndvi-normalized-difference-vegetation-index/

[7] Google. 2023. ERA5-Land Hourly - ECMWF Climate Reanalysis bookmark. Retrieved January 20, 2023 from https://developers.google.com/earth-engine/datasets/catalog/ECMWF_ERA5_LAND_HOURLY

[8] Google. 2023. MOD13Q1.061 Terra Vegetation Indices 16-Day Global 250m. Retrieved January 20, 2023 from https://developers.google.com/earth-engine/datasets/catalog/MODIS_061_MOD13Q1

[9] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. 1998. CURE: An efficient clustering algorithm for large databases. *ACM Sigmod record* 27, 2 (1998), 73–84.

[10] Perry Hystad, Ofer Amram, Funso Oje, Andrew Larkin, Kwadwo Boakye, Ally Avery, Assefaw Gebremedhin, and Glen Duncan. 2022. Bring your own location data: use of Google smartphone location history data for environmental health research. *Environmental Health Perspectives* 130, 11 (2022), 117005.

[11] George Karypis, Eui-Hong Han, and Vipin Kumar. 1999. Chameleon: Hierarchical clustering using dynamic modeling. *computer* 32, 8 (1999), 68–75.

[12] Leonard Kaufman and Peter J Rousseeuw. 1990. Partitioning around medoids (program pam). *Finding groups in data: an introduction to cluster analysis* 344 (1990), 68–125.

[13] Leonard Kaufman and Peter J Rousseeuw. 2009. *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, .

[14] Alexander Kmoch, Oleksandr Matsibora, Ivan Vasilyev, and Evelyn Uuemaa. 2022. Applied open-source Discrete Global Grid Systems. *AGILE: GIScience Series* 3 (2022), 41.

[15] Kisung Lee, Ling Liu, Raghu K Ganti, Mudhakar Srivatsa, Qi Zhang, Yang Zhou, and Qingyang Wang. 2016. Lightweight indexing and querying services for big spatial data. *IEEE Transactions on Services Computing* 12, 3 (2016), 343–355.

[16] Qing Li, Xue Li, Zuyu Liu, and Yaping Qi. 2022. Application of clustering algorithms in the location of electric taxi charging stations. *Sustainability* 14, 13 (2022), 7566.

[17] Bingxian Lin, Liangchen Zhou, Depeng Xu, A-Xing Zhu, and Guonian Lu. 2018. A discrete global grid system for earth system modeling. *International Journal of Geographical Information Science* 32, 4 (2018), 711–737.

[18] Eden WM Ma and Tommy WS Chow. 2004. A new shifting grid clustering algorithm. *Pattern recognition* 37, 3 (2004), 503–514.

[19] J MacQueen. 1967. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*. University of California Los Angeles LA USA, ., ., 281–297.

[20] T Soni Madhulatha. 2012. An overview on clustering methods. *arXiv preprint arXiv:1205.1117* ., . (2012), .

[21] Burkhard Monien, Robert Preis, and Stefan Schamberger. 2007. Approximation algorithms for multilevel graph partitioning. *Handbook of approximation algorithms and Metaheuristics* 10 (2007), 1–60.

[22] Raymond T. Ng and Jiawei Han. 2002. CLARANS: A method for clustering objects for spatial data mining. *IEEE transactions on knowledge and data engineering* 14, 5 (2002), 1003–1016.

[23] Hae-Sang Park and Chi-Hyuck Jun. 2009. A simple and fast algorithm for K-medoids clustering. *Expert systems with applications* 36, 2 (2009), 3336–3341.

[24] Matthew BJ Purss, Robert Gibb, Faramarz Samavati, Perry Peterson, and Jin Ben. 2016. The OGC® Discrete Global Grid System core standard: A framework for rapid geospatial integration. In *2016 IEEE international geoscience and remote sensing symposium (IGARSS)*. IEEE, ., ., 3610–3613.

[25] Peter Sanders and Christian Schulz. 2011. Engineering multilevel graph partitioning algorithms. In *Algorithms–ESA 2011: 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings 19*. Springer, ., ., 469–480.

[26] Erich Schikuta. 1996. Grid-clustering: An efficient hierarchical clustering method for very large data sets. In *Proceedings of 13th international conference on pattern recognition*, Vol. 2. IEEE, ., ., 101–105.

[27] Horst D Simon. 1991. Partitioning of unstructured problems for parallel processing. *Computing systems in engineering* 2, 2-3 (1991), 135–148.

[28] WenBin Sun, MaJun Cui, XueSheng Zhao, and YanLi Gao. 2008. A global discrete grid modeling method based on the spherical degenerate quadtree. In *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing*, Vol. 2. IEEE, ., ., 308–311.

[29] Iping Supriana Suwardi, Dody Dharma, Dicky Prima Satya, and Dessi Puji Lestari. 2015. Geohash index based spatial data model for corporate. In *2015 International Conference on Electrical Engineering and Informatics (ICEEI)*. IEEE, ., ., 478–483.

[30] J Swarndeep Saket and Sharnil Pandya. 2016. An overview of partitioning algorithms in clustering techniques. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* 5, 6 (2016), 1943–1946.

[31] Uber. 2024. H3 - Uber's Hexagonal Hierarchical Spatial Index - Uber Blog. Retrieved January 24, 2024 from https://www.uber.com/blog/h3/

[32] Uber. 2024. Overview of the H3 Geospatial Indexing System. Retrieved January 24, 2024 from https://h3geo.org/docs/core-library/overview/

[33] Chris Walshaw. 2004. Multilevel refinement for combinatorial optimisation problems. *Annals of Operations Research* 131 (2004), 325–372.

[34] Yitian Wu, Gang Wan, Lei Liu, Yao Mu, Zhanji Wei, and Shuai Wang. 2022. A Review of the Research on Discrete Global Grid Systems in Digital Earth. In *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, Vol. 10. IEEE, ., ., 1974–1978.

[35] Dongkuan Xu and Yingjie Tian. 2015. A Comprehensive Survey of Clustering Algorithms. *Annals of Data Science* 2, 2 (01 Jun 2015), 165–193. https://doi.org/10.1007/s40745-015-0040-1

[36] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: an efficient data clustering method for very large databases. *ACM sigmod record* 25, 2 (1996), 103–114.

[37] JH Zhao, XZ Wang, FY Wang, ZH Shen, YC Zhou, and YL Wang. 2017. A novel approach of indexing and retrieving spatial polygons for efficient spatial region queries. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 4 (2017), 131–138.