Clustering-based compression for raster time series

Martita Muñoz[1,2,3], José Fuentes-Sepúlveda[1,2], Cecilia Hernández[1,3], Gonzalo Navarro[2,3,4], Diego Seco[5] and Fernando Silva-Coira[5]

[1]*Department of Computer Science, Universidad de Concepción, Chile*
[2]*Millennium Institute for Foundational Research on Data, Chile*
[3]*Center of Biotechnology and Bioengineering (CeBiB), Chile*
[4]*Department of Computer Science, Universidad de Chile, Chile*
[5]*CITIC, Facultade de Informática, Universidade da Coruña, Spain*
Email: martitamunoz@udec.cl

A raster time series is a sequence of independent rasters arranged chronologically covering the same geographical area. These are commonly used to depict the temporal evolution of represented variables. The T-k^2-raster is a compact data structure that performs very well in practice for compact representations for raster time series. This structure classifies each raster as a snapshot or a log and encodes logs concerning their reference snapshots, which are the immediately preceding selected snapshots. An enhanced version of the T-k^2-raster, called Heuristic T-k^2-raster, incorporates a heuristic for automating the selection of snapshots. In this study, we investigate the optimality of the heuristic employed in Heuristic T-k^2-raster by comparing it with a dynamic programming approach. Our experimental evaluation demonstrates that Heuristic T-k^2-raster is a near-optimal solution, achieving compression performance almost identical to the dynamic programming method. These results indicate that variations of the structure that maintain the temporal order of the rasters are unlikely to significantly improve compression. Consequently, we explore an alternative approach based on clustering, where rasters are grouped according to their similarity, regardless of their temporal order. Our experimental evaluation reveals that this clustering-based strategy can enhance compression in scenarios characterized by cyclic behavior.

Keywords: Raster dataset; Temporal Raster; Data compression; Compact Data Structure; Clustering; Dynamic Programming

1. INTRODUCTION

The raster data model is a structured representation of data consisting of a regular grid of square cells, where each cell contains a value defined by the modeled data. The raster model is often used in Geographic Information Systems (GIS) [1, 2] because it is well-suited to represent natural phenomena, such as terrain elevation, humidity, atmospheric pressure, and temperature, distributed over geographic space [3]. A related model is the raster time series, a.k.a. temporal raster, a time-ordered sequence of discrete, independent rasters covering the same geographic space. The principal utility of this model is in representing the temporal evolution of the variables defined in each raster. This model is used in various fields where tracking changes in spatially-distributed variables over time is critical [3]. In addition to GIS applications, the raster data and raster time series models have applications in image analysis, including images from medical [4, 5, 6], astronomical [7] and hyperspectral [8, 9, 10] domains. By providing a flexible and robust framework for representing and analyzing data, these models have become valuable tools for researchers across various disciplines, facilitating the investigation of complex phenomena and developing new insights and understanding.

Large volumes of raster data and raster time series are currently available. For instance, advances in remote sensing and instrumentation in geospatial sensors and satellites have enabled the acquisition of vast amounts of information at high frequency and resolution [11], rapidly increasing the volume and size of tracking data [12]. The modern remote imaging sensors collect and manage among terabyte-scale and zettabyte-scales amounts of Earth observation images [13, 14]. The enormous amount of data generated by geospatial sensors and satellites presents significant challenges in data processing, management, and analysis.

One of the key features of raster data is the data

locality. Nearby cells, spatially and/or temporally, tend to exhibit similar or slightly different values. This property has been leveraged to develop various data compression techniques, such as compact data structures (CDS) [15]. CDS are specialized data structures designed to represent different data types, including raster data, compactly or succinctly, with query support without decompression. By exploiting the spatial and temporal coherence in the data, CDS can achieve significant compression ratios while maintaining efficient query support [15].

There are several proposed compact representations for raster and temporal rasters [16, 17, 18, 19, 12, 3, 20]. These representations are built on top of the k^2-tree [21, 22, 23], designed specifically for sparse binary matrices, and performing a similar partition than the well-known Quadtree data structure [24]. This structure recursively subdivides the matrix in submatrices, stopping when it finds submatrices with the same value or individual cells. The k^2-raster [19, 12] is a compact representation of raster data, which is a prominent structure that operates similarly to the k^2-tree. In the k^2-raster, the minimum and maximum values of each submatrix are stored to enable efficient value lookup at each location. The T-k^2-raster is a compact data structure that builds upon the k^2-raster to represent raster time series [3, 20]. In this structure, each individual raster is classified as either a **snapshot** or a **log** and represented using a variant of the k^2-raster. A snapshot stores the original values, while a log stores the differences between the current raster and the previously generated snapshot. The T-k^2-raster selects snapshots at fixed time intervals to capture the temporal evolution of the raster data. A variant of T-k^2-raster is the Heuristic T-k^2-raster, which includes a heuristic for automatically selecting snapshots and logs [3]. This structure leverages the heuristic to improve the snapshot selection efficiency and reduce storage requirements. The Heuristic T-k^2-raster has been used in various applications, including remote sensing and cartography, where efficient management and analysis of large-scale temporal rasters data are critical requirements. Based on experimental results, the compression space and query time of the T-k^2-raster and the Heuristic T-k^2-raster are competitive [3]. However, both compact representations have a constraint regarding the snapshot selection. For a raster log, only the latest previous selected snapshot in chronological order can be its respective snapshot reference. This limitation impede selecting the best snapshot candidate to improve the data representation. This is important in domains where the variable under study exhibits cyclic behavior, meaning that the values represented at one point in time can repeat or be very similar at other points in time (e.g. the temperature at a specific hour of the day may be similar on other days during the same season). In other words, applying clustering to the T-k^2-raster allows us to exploit not only spatial and temporal locality but also the cyclic property.

This work presents a comprehensive study of the heuristic used for selecting snapshots in the Heuristic T-k^2-raster. To evaluate the effectiveness of the proposed heuristic, we compare it with other snapshot selection strategies aimed at improving the compression ratio. Specifically, we use dynamic programming to determine the optimal selection of snapshots and compare its compression performance with the proposed heuristic. It is relevant to indicate that dynamic programming presents the same snapshot selection constraint that T-k^2-raster. Our experimental evaluation shows that the Heuristic T-k^2-raster is near optimal since it achieves almost the same compression performance as the dynamic programming solution. Therefore, to further reduce the space usage of the data structure, we explore alternative representations that eliminate the snapshot selection constraint. Specifically, we explore the application of clustering algorithms using a distance measure based on Hamming distance. Using clustering enables us to choose as snapshots those rasters that are the centroids of the clusters, and therefore the representation is not restricted to following the time-ordered rasters. However, to enable the same query support of the T-k^2-raster, an additional integer vector is needed to identify, for each raster, its respective snapshot or cluster centroid. Our results show that clustering can improve the compression performance of the Heuristic T-k^2-raster when the raster time series shows cyclic behavior, keeping the query support performance.

The article is structured as follows. Section 2 provides essential background information, introducing key concepts necessary for understanding the study. In Section 3, relevant related work is discussed. Section 4 explains the application of dynamic programming to the T-k^2-raster. Section 5 elaborates on the application of clustering to the T-k^2-raster. The experimental process, results, and corresponding discussion are presented in Section 6. Finally, Section 7 presents the conclusions drawn from this study and outlines potential future directions.

2. BACKGROUND

This section provides an overview of the background that helps to understand our study. The principal subjects are the compact data structures and the Clustering techniques, revised in Sections 2.1 and 2.2, respectively.

2.1. Compact data structures (CDS)

A compact data structure (CDS) is a data structure that represents different types of data (trees, tables, sets, graphs, text, among others) using a small amount of space, close to the minimum indicated by information theory. Furthermore, these structures can efficiently support required operations on their data [15].

One of the essential CDS is the bitmap or bit array, which offers two crucial queries: *rank* and *select* [15]. The rank query returns the number of symbols, either zeros or ones, within a bitmap up to a specified position i. In contrast, the select operation returns the position of the i-th symbol within a bitmap [15]. The bitmap data structure is a fundamental building block for many other CDS.

A relevant CDS that inspired the k^2-raster and the T-k^2-raster is the k^2-tree. The k^2-tree is a CDS that was initially designed to represent Web graphs as a compact representation of their adjacency matrix; however, it has also been applied to compress sparse binary matrices in different domains [21, 22, 23]. The k^2-tree is based on the idea of Quadtrees [24] and reduces space consumption by compacting submatrices that are full of zeros.

Given an $n \times n$ matrix, where n is a power of k,[1] the matrix is subdivided into k^2 submatrices of $\frac{n}{k} \times \frac{n}{k}$, which are counted from left to right and top to bottom. Each submatrix is represented by a bit whose value is one if the submatrix contains at least one cell with a value of one, and zero if all cells are zero. The submatrices represented with a value of one are recursively subdivided into k^2 submatrices, and the subdivision continues until the submatrix contains only zeros or is an individual cell. The recursive subdivision of the matrix generates a conceptual tree that is stored following a level order traversal, storing the bits produced by the k^2-tree. Common implementations of the k^2-tree use two bitmaps to store the traversal: T, which stores the tree node values except the last level, and L, which stores the last level tree node values. Tree navigation can be efficiently implemented using the operations described earlier for bitmaps.

Other CDS more directly related to our work, such as the k^2-raster and its generalization to temporal rasters, are described in Section 3.

2.2. Clustering

Clustering is a technique that aims to group elements, referred to as "points", into clusters based on a distance metric. The goal is to create groups where each group contains close points, and the distance between points in different groups is high. Clustering is fundamental in many fields, such as machine learning, data mining, and pattern recognition enabling the discovery of hidden patterns, performing exploratory data analysis, and reducing the dimensionality of large datasets [25]. Some well-known distance metrics include Euclidean, Hamming, Edit, and Jaccard distances, where choosing an appropriate distance metric is crucial to obtain meaningful and effective clustering results [26].

Two popular clustering strategies are hierarchical and partition-based clustering. Hierarchical clustering, or agglomerative clustering [26], starts by defining each data point as a separate cluster. Next, it proceeds to iteratively merge the two closest clusters into a single larger cluster until the desired number of clusters is reached. The hierarchical clustering methods can be classified based on how they compute the distance metric. Some commonly used methods include Single-link clustering, Complete-link clustering, and Average-link clustering [26]. In this study, we apply the Complete-link scheme, which measures the similarity between two clusters by considering the maximum distance between any two points, one from each cluster.

On the other hand, the partition-based clustering scheme employed in this study is the k-means algorithm [27, 28]. The algorithm starts by selecting k representative points as the initial centroids of the k clusters. Subsequently, the algorithm assigns each non-representative point to the closest cluster based on the distance between the point and the cluster centroid. Since including new points affects the cluster centroid, the algorithm performs multiple iterations over the entire set of points until they remain in the assigned cluster. The number of iterations can be set to a fixed number or performed until the algorithm converges. The selection of the initial k centroids is a crucial aspect of the k-means algorithm, as it can significantly impact its effectiveness. Typically, the initial k points are selected randomly, but this approach can be problematic if two points too close to each other are selected. The k-means++ is a variant of the k-means algorithm to reduce the effect of randomness in selecting the k points that represent the clusters [29]. The method biases the selection by randomly choosing the first representative point from the data points and then selecting the following points with probability proportional to the square of their distance from the nearest already chosen center. This approach helps to improve the selection of the first k points by choosing points farther apart. Despite the additional initialization cost, the k-means++ algorithm converges faster and produces better results than the standard k-means algorithm, reason why we use it in our study.

In addition, the effectiveness of many clustering techniques also depends on selecting the number of clusters. Several indices have been proposed to estimate the number of clusters, including the Silhouette index, which compares the intracluster and intercluster distances of the partitioning and it is defined as follows [30]: Given a point p_i, compute the intracluster distance $a(i) = \frac{1}{|C_I|-1}\Sigma_{j \in C_I, i \neq j} d(i,j)$ and the intercluster distance $b(i) = \min_{J \neq I}\{\frac{1}{|C_J|}\Sigma_{j \in C_J} d(i,j)\}$, where C_I is the cluster that contains p_i, C_J is another cluster and $d(i,j)$ is the distance applied to points p_i and p_j. Finally, the Silhouette value of point p_i, named $s(i)$, can be computed by applying Equation 1:

[1] If the matrix is non-square or n is not a power of k, the k^2-tree structure extends to the smallest power of 2 greater than n.

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad (1)$$

The Silhouette value assesses the classification quality of an individual data point within its cluster, while the Silhouette index S represents the average of these values across all points in a clustering solution and estimates the clustering quality. The Silhouette index ranges between -1 and 1, where a higher value indicates that the points are well-clustered. Hence, the ideal number of clusters k is such that maximizes the average Silhouette index.

3. RELATED WORK

This section provides an overview of the related work on representing raster data and raster time series. Specifically, we present the functionalities of two principal compact data structures (CDS): the k^2-raster and the T-k^2-raster. Additionally, we describe related applications of clustering techniques to raster data.

3.1. k^2-raster

The k^2-raster is a CDS that represents raster data succinctly based on the k^2-tree representation [19, 12]. Its construction involves the recursive subdivision of an $n \times n$ matrix, where n is a power of k,[2] into a conceptual tree such that each node represents a submatrix using a subarray storing its respective submatrix's minimum and maximum values. The subdivision process continues until all values within a subarray are identical or individual cells are reached.

However, to improve the compression of the k^2-raster structure, it does not store absolute minimum and maximum values. Instead, it stores the difference between each value and its equivalent in the parent node. These differences are efficiently represented with bit encoders, particularly with Directly Addressable Codes (DACs) [31]. As a result, only the root node stores the absolute minimum and maximum values. This approach reduces the size of the values ultimately stored in the structure, leading to improved compression. The k^2-raster structure allows for easy retrieval of original values, as the query process involves traversing the tree from root to leaf, accumulating the stored differences along the path.

Figure 1 presents a complete example of a k^2-raster representation. The structure recursively subdivides the matrix into four submatrices, and each generated submatrix's minimum and maximum values are represented in the tree. If a submatrix contains only equal values, the subdivision process terminates for that submatrix, as seen in the lower right submatrix of the example, where all the cells contains the value 2. However, for the remaining submatrices, the recursive subdivision continues.

The next step is to compute the differences between the values of the nodes and their parent's nodes values. As a result, it returns the conceptual tree representation presented on the bottom left part of the figure. The root node stores the global maximum and minimum values of 5 and 1, respectively. The first submatrix in the top left corner contains 5 and 3, corresponding to the range of values inside the submatrix. The difference between the maximum value of the submatrix and that of its parent node is 0, whereas the difference between the respective minimum values is 2. Therefore, the submatrix stores the values of 0 and 2 as its maximum and minimum values, respectively, using this approach.

The main components used for the implementation of the k^2-raster are the following:

- $rMin$ and $rMax$: These variables correspond to the minimum and maximum values of the entire matrix, respectively.
- $Lmin$ and $Lmax$: These structures represent the minimum and maximum values, respectively, that are obtained along the level-order traversal of the conceptual tree. Note that the values of the last level are only represented in $Lmax$. Both structures are stored using Directly Addressable Codes (DACs) [31].
- $Tree$: This bitmap structure represents the topology of the k^2-raster tree and works similarly to the k^2-tree.

The authors present two variants of the k^2-raster structure in their work. The first variant, referred to as k^2-raster$_H$, uses two values of k, denoted as k_1 and k_2 [19]. Along the initial n_1 levels, the structure uses a value of $k = k_1$, while on the subsequent levels, the structure uses a value of $k = k_2$. The second variant of the k^2-raster proposed by the authors is the k^2_H-raster [12]. This variant employs an entropy-based heuristic approach to represent the last levels of $Lmax$, where the most frequent values are encoded using shorter codewords. Recursive subdivision stops at the last l levels, resulting in submatrices of size $k_{lst} \times k_{lst}$. Each submatrix generated is assigned a code based on its frequency of appearance, creating a vocabulary that associates each submatrix with a unique code. The structure then selects whether to represent each submatrix of the dictionary using the assigned code or its original values, depending on the size required to represent the submatrices.

3.2. T-k^2-raster

The T-k^2-raster is a CDS designed to represent a raster time series efficiently [3, 20]. The structure employs the k^2-raster to represent each raster in the temporal raster. The CDS defines two types of rasters: snapshots and logs. Snapshots are taken at fixed intervals and serve

[2] If the matrix is non-square or n is not a power of k, the k^2-raster structure extends to the smallest power of 2 greater than n, similarly to the k^2-tree.

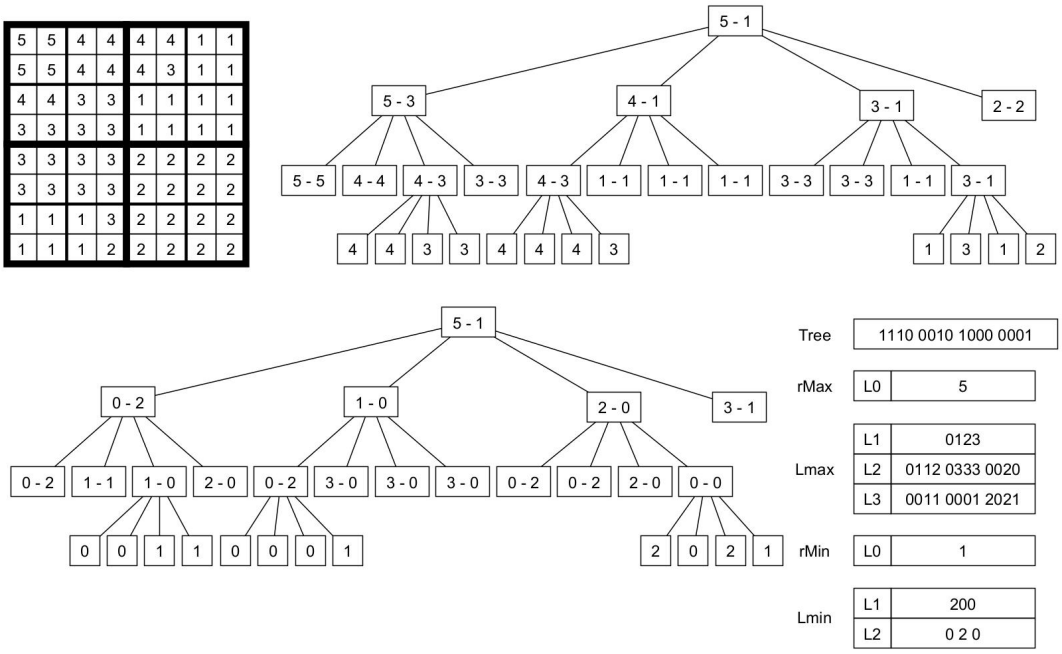


FIGURE 1: An example of a k^2-raster. On the top, the figure shows an 8×8 raster example and its respective conceptual tree representation using $k = 2$. On the bottom, it is shown the conceptual tree of differences and the final components of the data structure.

as a reference for representing subsequent raster logs. The raster logs store the differences between the values the log represents and the reference snapshot used. The logs are created by considering the most recent snapshot as a reference.

T-k^2-raster uses a modified version of the k^2-raster, namely k^2-raster$_{log}$, to represent logs. In contrast to the regular k^2-raster, which is used to represent snapshots, the k^2-raster$_{log}$ requires an additional bitmap called eqB to differentiate cases when a submatrix is not subdivided. In the case of a log raster, this occurs when all the values in the submatrix are equal (as in the standard case of the k^2-raster) or when the values between the submatrix and its equivalent in the snapshot vary by a constant.

Figure 2 presents a T-k^2-raster example representing a raster time series with two rasters. The first raster is a snapshot, while the second is a log referenced to its predecessor. The first raster employs a k^2-raster, while the second raster employs a k^2-raster$_{log}$. The second conceptual tree reflects in its nodes the two cases that differentiate eqB: the standard case of the k^2-raster and when the values between the submatrix and its equivalent in the snapshot vary by a constant.

The Heuristic T-k^2-raster is a variant of the T-k^2-raster that incorporates a heuristic approach to improve the space usage of the data structure by selecting *more suitable* snapshots [3]. Unlike the regular T-k^2-raster, the distance between snapshots is not necessarily a fixed value. Therefore, the structure includes a bitmap that allows to identify which rasters are snapshots and which are logs. The Heuristic T-k^2-raster features a heuristic algorithm that iterates through each raster and evaluates three possible scenarios. The algorithm determines the structure's size (in bytes) up to the current iteration and selects the case that results in the smallest size. The three considered cases are: (1) the selected raster is a snapshot, (2) the previous raster is converted to a snapshot, and the selected raster is a log of this new snapshot and (3) the selected raster is a log of the last defined snapshot.

The proposed heuristic has certain limitations. The first is that the heuristic restricts the inclusion in the snapshot set of rasters that happened before the previous raster in revision, and the heuristic does not allow for removing rasters from the snapshot set. The second limitation is the selection of logs that a snapshot can reference. Specifically, exclusively the most recently generated snapshot in the temporal order can reference a log. This constraint may limit the ability of the heuristic to identify the optimal selection of snapshots to improve the compression.

3.3. Applications of clustering to raster data

There are several works related to the application of clustering on raster data. Alkathiri et al. [32, 33] investigated the utilization of k-means clustering for processing multi-spectral geo-spatial raster data in a Hadoop environment. Alzaghoul et al. [34] applied clustering to rasters representing Digital Elevation Models, aiming to identify hidden patterns, uncover

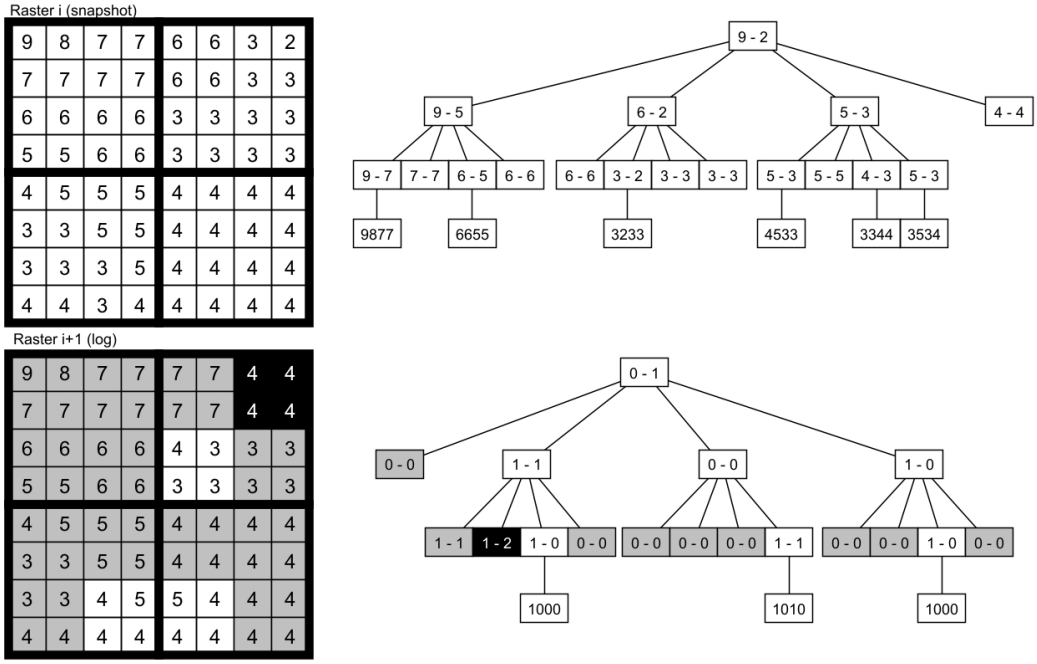


FIGURE 2: A T-k^2-raster example with two rasters of size 8×8. Both rasters are accompanied by their respective conceptual tree. The gray and black submatrices/nodes in the second raster represent the two cases that differentiate eqB. Black submatrices/nodes reflect the standard case of the k^2-raster when all the values in the submatrix are equal. Gray submatrices/nodes represent when the values between the submatrix and its equivalent in the snapshot vary by a constant.

relationships, and discover clusters of elevation values. Image compression of RGB photos was addressed by authors in [35], where clustering algorithms were employed. Kiran [36] applied clustering to discover knowledge from raster data. Mariani et al. [37] presented a distributed clustering algorithm to handle big data rasters in a decentralized manner. Aghaee et al. [38] applied clustering to predict geological lineaments using topographic, magnetic, and gravity raster data. Wu et al. [39] proposed a pixel clustering-based method to enhance the efficiency of mining spatial sequential patterns from raster serial remote sensing images (SRSI). These studies demonstrate the wide range of applications and the potential benefits of employing clustering techniques in the analysis and processing of raster data.

Sisodiya et al. [40] applied clustering on raster data compacted in a k^2-raster. The authors aimed to overcome memory limitations associated with traditional clustering methods when dealing with large datasets containing raster values. The findings of this study highlight the potential of employing the k^2-raster and clustering methods to analyze raster data in a more efficient and scalable manner. Their research demonstrated that the proposed approach, based on a CDS, effectively addressed the challenges of data representation and scalability in clustering.

4. AN OPTIMAL T-k^2-RASTER VIA DYNAMIC PROGRAMMING

Section 3.2 presents an overview of the functioning and representation of a temporal raster using the Heuristic T-k^2-raster. The approach classifies rasters as either snapshots or logs. As the heuristic reviews each raster in temporal order, it can only select the current or the previous raster as a new snapshot. This section describes a dynamic programming (DP) algorithm to select the optimal subset of rasters as snapshots that minimizes the space usage of the T-k^2-raster for the temporal raster. For each raster in the input sequence, this approach decides if a raster is a log or a snapshot exploring all previously defined snapshots and not only the previous or last snapshots, as used by the T-k^2-raster.

Let $\mathcal{M}$ be a raster time series of τ raster time instants, in which each raster is of size $n \times n$. The aim is to determine a snapshot subset $\mathcal{M}_s$ of $\mathcal{M}$ that represents the entire raster time series using a T-k^2-raster with minimal storage space required. Here, S represents the sorted index set of the selected subset ($\forall i \in S, i \in [1, \tau]$). The selection of this subset $\mathcal{M}_s$ is crucial for capturing the essential information of the raster time series while optimizing storage efficiency. For this, it is necessary to define the following operations:

- $ref(i)$: This operation calculates the space required to represent the raster M_i using a k^2-

raster.

- $c(i,j)$: This operation calculates the space required to represent the raster M_i as a k^2-raster$_{log}$ using M_j as a reference.

The proposed approach identifies the subset $\mathcal{M}_s$ by solving an optimization problem using DP. Specifically, the goal is to find the index subset S that minimizes the following function:

$$\sum_{j=1}^{j\leq|S|}\left[ref(S[j]) + \sum_{i=S[j]+1}^{i<S[j+1]} c(i,S[j])\right] \quad (2)$$

where $S[|S|+1]-1=\tau$ for convenience.

Figure 3a illustrates an example of the application of DP over a particular raster time series. In the implementation, the structure indicates the raster snapshots with a bitmap s_bv, i.e. $s\_bv[i]$ is 1 if raster i is a snapshot. cs_v vector stores for each raster the position of its respective snapshot. For rasters snapshots stores the same raster position. This structure is relevant for clustering application (see Section 5), and now is introduced only for illustrative purpose. Note that cs_v is not actually stored as it can be computed from $s\_bv[i]$ (i.e. $cs\_v[i] = select(rank(s\_bv[i]))$). In the example, the first raster time instant, M_1, is selected as a snapshot by default. Rasters M_2 and M_3 are encoded using M_1 as a reference, enabling the representation of the first three raster time instants using minimal space. As the difference between M_4 and M_1 is considerable, DP selects M_4 as a new snapshot. Besides, the difference between rasters M_4 and M_8 is considerable, enabling M_8 to be selected as a new snapshot. Hence, the selected subset of rasters is $\mathcal{M}_s = \langle M_1, M_4, M_8, M_9\rangle$, with the corresponding sorted index set $S=\{1,4,8,9\}$, and $|S|=4$.

Let $\mathcal{M}[i,\ldots,j]$ denote a subproblem considering the raster intervals from M_i to M_j. We have two alternatives for selecting a snapshot. Firstly, we can consider M_i as the only snapshot where the space representation of the subproblem is minimal, which serves as the base case. The second option is to identify a position $r \in [i+1, j]$ such that M_r represents the subproblem using the minimum space. In the case that DP chooses the second option and selects the snapshot M_r, the problem can be subdivided into two subproblems: $\mathcal{M}[i,\ldots,r-1]$ and $\mathcal{M}[r,\ldots,j]$, which can be solved recursively.

Equation 3 defines the optimization problem that needs to be solved for the subproblem.

$$\min\left(\left(ref(i) + \sum_{k=i+1}^{j} c(k,i)\right), \min_{i<r\leq j}\left[ref(i)+ \sum_{k=i+1}^{r-1} c(k,i) + ref(r) + \sum_{k=r+1}^{j} c(k,r)\right]\right) \quad (3)$$

Consider a matrix $\mathbb{M}$ with dimensions of $\tau\times\tau$, where $\mathbb{M}[i,j]$ represents the minimum space required to represent the subproblem $\mathcal{M}[1,\ldots,i]$ using M_j as the last snapshot, with $j \leq i$. The matrix $\mathbb{M}$ can be computed using the recursive equation 4, with the calculation of each cell performed in row-major order from row 1 to τ, and from cell $[i,1]$ to $[i,i]$ inside each row i.

$$\mathbb{M}[i,j] = \begin{cases} \min\limits_{1\leq k\leq i-1} \mathbb{M}[i-1,k] + ref(i) & \text{if } i=j, \\ \min\limits_{1\leq k\leq j} \mathbb{M}[j-1,k] + ref(j) + \sum\limits_{k=j+1}^{k\leq i} c(k,j) & \text{if } j<i \end{cases} \quad (4)$$

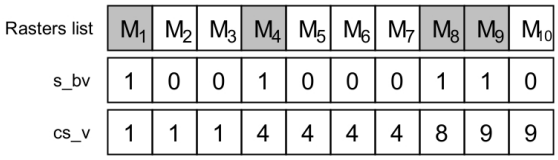
In the first part of Equation 4, the last raster is assumed to be a snapshot. To achieve this, the minimum space required to represent the raster time instants $\mathcal{M}[1,\ldots,i-1]$, along with the space required to represent M_i as a snapshot, is calculated. The second part of the equation assumes that M_j is the last snapshot. To compute the minimum space required for this case, the space needed to represent the raster time instants $\mathcal{M}[1,\ldots,j-1]$ is added to the space required to represent M_j as a snapshot, plus the space required to represent the raster time instants $\mathcal{M}[j+1,\ldots,i]$ encoded using M_j as a reference.

To compute the set of indexes S corresponding to the selected snapshots, we need to iterate through each row of the matrix $\mathbb{M}$ in reverse order, starting from the last row τ. For each row i, we compute the set of indexes $\{s_i | \min_{1\leq s_i\leq i} \mathbb{M}[i,s_i]\}$, which correspond to the snapshots that minimize the space required to represent the subproblem $\mathcal{M}[1,\ldots,i]$. We collect all the different indexes s_i for each row i and store them in the set S. Finally, the selected snapshots are the set of rasters $\{M_s | s\in S\}$. The minimum space required to represent the entire $\mathcal{M}$ using the selected snapshots is given by $\min_{1\leq j\leq\tau}\mathbb{M}[\tau,j]$.

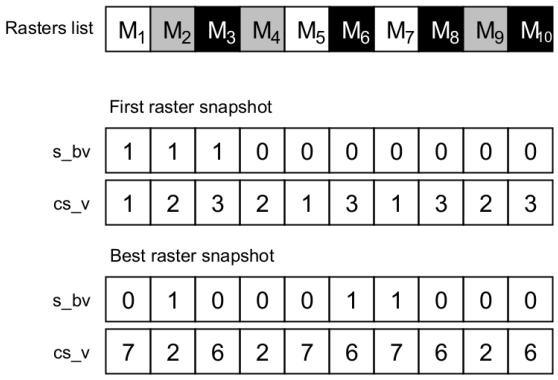
The example in Figure 4 illustrates the process of selecting the last snapshot over the raster time series $\mathcal{M}$ to minimize its space representation. The black cells are not used because i is not less than j. The gray cells indicate the last snapshot selected for each row i that minimizes the space representation of $\mathcal{M}[1,\ldots,i]$. For instance, for $i\in[1,3]$, the last snapshot selected is 1. However, for $i=4$, the last snapshot selected is 4 as it minimizes the space required to represent $\mathcal{M}[1,\ldots,4]$.

5. USING CLUSTERING TO IMPROVE THE T-k^2-RASTER

In Sections 2 and 4, we discussed different approaches for selecting snapshots to succinctly represent a raster time series using a T-k^2 raster. A common characteristic of such techniques is that a raster log uses as a reference a preceding raster snapshot



(a) A raster time series example with their respective snapshots and logs rasters. The gray cells in the list of rasters represent snapshots, while the white cells represent logs. cs_v is only represented for illustrative reasons. cs_v can be calculated as $cs\_v[i] = select(rank(s\_bv[i]))$.



(b) Clustering application example. The gray cells in the list of rasters represent the rasters belonging to a first cluster, the white cells represent the rasters belonging to a second cluster, and the black cells represent the rasters belonging to a third raster. s_bv is only represented for illustrative reasons. s_bv can be calculated as $s\_bv[i] = cs\_v[i] == i$.

FIGURE 3: An example of the application of dynamic programming and Clustering on a raster time series with ten raster instants ($\tau = 10$). Each raster is represented as a cell in the Rasters list. Bitmap s_bv marks the rasters selected as snapshots. Vector cs_v shows the raster snapshot referenced by each raster.

in time-order. However, this limitation restricts exploring alternative combinations, such as referencing a subsequent snapshot that may be more similar than any of the previous ones. By considering these alternative strategies, it becomes possible to discover new combinations of snapshots and logs that can effectively minimize the size of the data structure.

Depending on the temporal locality, the distance between neighboring or nearby rasters is expected to be less than between distant rasters. However, in certain cases, this pattern may not hold or additional patterns may also exist. For instance, if the raster time series reflects a cyclical variable, the values may repeat every certain number of rasters. Therefore, it is crucial to identify such patterns and group similar rasters together, regardless of the number of rasters that separate them.

To address this, we introduce the application of the clustering technique to enable the selection of snapshots that correspond to each log. This process aims to reduce the size of the T-k^2 raster used to represent the data.

For the application of clustering analysis to our raster time series problem, individual rasters are considered as "points". The order of the rasters within the time series is disregarded, allowing them to be rearranged and grouped according to their similarity. This approach enables us to focus on the similarities among the rasters and disregard their temporal dependence.

While storing the referenced snapshot for each log is necessary to achieve a complete representation of a T-k^2 raster, the additional space required for this purpose is insignificant. Each raster in the temporal raster requires a constant value to indicate its corresponding snapshot. In Figure 3b, cs_v stores these values. In the case of a snapshot raster, the value will point to itself, indicating that it is a snapshot. For this reason, s_bv is not necessary to indicate snapshot selected rasters. s_bv can be computed from cs_v (i.e. $s\_bv[i] = cs\_v[i] == i$). Therefore, the storage overhead associated with storing snapshot references is minimal and does not significantly impact the overall size of the representation.

In the rest of this section, we describe the relevant configuration steps required for the application of clustering. First, Section 5.1 presents the distance measures applied in this study. Then, Section 5.2 describes the selection of the number of clusters. Finally, Section 5.3 details the selection of a raster centroid for each cluster, which is then represented as a snapshot whereas all the other rasters in the cluster are represented as logs with respect to such a snapshot.

5.1. Distance measures

The choice of a suitable distance measure is crucial to compare a set of rasters, as it must be sensitive to the differences between any pair of rasters. If the two rasters are identical, meaning that the values of all their cells are the same, the distance should be zero. As the differences between the rasters increases, the distance metric should also increase accordingly. To define a distance measure that effectively captures the differences between two rasters, we consider two criteria: (1) the number of cells that differ between the rasters and (2) the magnitude of the differences between those cells. By incorporating both criteria into the

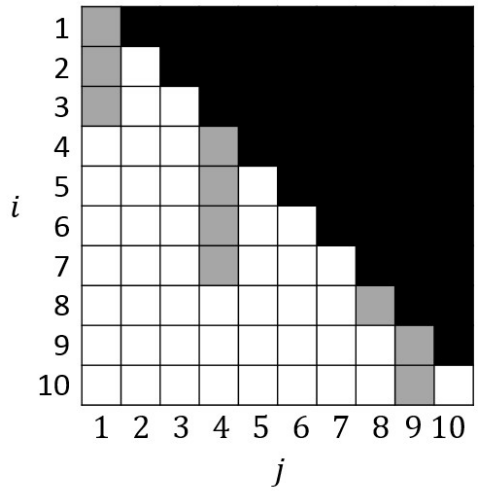


FIGURE 4: Matrix $\mathbb{M}$ example from Figure 3a. To compute the entries of row $\mathbb{M}[6]$ of the matrix, we evaluate for each column $i \in [1,6]$, the minimum space required to represent $\mathcal{M}[1,\ldots,6]$ using M_i as the last snapshot. For example, to compute $\mathbb{M}[6,3]$, we first determine the minimum space required to represent $\mathcal{M}[1,2]$ using any snapshot k such that $1 \leq k \leq 2$ $(\min_{1\leq k\leq 2}\mathbb{M}[2,k])$. We then add the space required to represent M_3 as a snapshot $(ref(3))$ and the space required to represent $\mathcal{M}[4,\ldots,6]$ encoded by M_3 $(\sum_{k=4}^{k\leq 6} c(k,3))$.

distance measure, we aim to establish a comprehensive measure that appropriately accounts for the variations between rasters.

We applied two distance measures based on the Hamming distance [41]. Let M and N be two matrices with $|M|$ and $|N|$ cells, respectively (where $|M| = |N|$), and let m_i and n_i denote the value of a cell in M and N, respectively.

The *Normalized Hamming distance* (NHD) [42, 43], a variant of the Hamming distance, is commonly used in decision making process, but its most basic application is to compare strings of equal length by counting the differing symbols. In the context of raster data, this distance measure can be adapted to quantify the dissimilarity between two rasters by counting the number of differing cells. Equation 5 presents the formula, where the count of differing cells is divided by the total number of cells in the raster. The resulting value ranges between 0 and 1, with values closer to 0 indicating more significant similarity between the compared rasters.

$$H = \frac{\sum_{0\leq i<|M|}(m_i \neq n_i)}{|M|} \tag{5}$$

Where H is the Normalized Hamming distance (NHD).

The *Weighted Hamming Distance* (WHD) [42, 43] is a second variant of the Hamming distance. It calculates the average of the absolute differences between all cells of two compared rasters (see Equation 6). The resulting value measures the dissimilarity between the rasters, with a larger H_w indicating more significant differences between them.

$$H_w = \frac{\sum_{0\leq i<|M|}|m_i - n_i|}{|M|} \tag{6}$$

Where H_w is the Weighted Hamming Distance (WHD).

The *Combined Hamming Distance* (CHD) is a distance measure that weighs the NHD and WHD. The formula for CHD can be expressed as shown in Equation 7.

$$H_c = H \times H_w \tag{7}$$

Where H_c is the Combined Hamming Distance (CHD).

Distance	$[2,2]$ **and** $[4,4]$	$[2,2]$ **and** $[6,2]$
H_w	2	2
H_c	2	1

TABLE 1: Example of computation of H, H_w and H_c distances measures over 1×2 matrices

Table 1 shows an example of the calculus of the different distance measures presented. In this example, the distances computed using H_w yield the same result for both computations, as the weight values are identical. However, H_c produces different results. This discrepancy arises because the two matrices in the first computation differ in two cells, while in the second computation, they differ in only one cell.

By combining H and H_w, H_c can capture the difference between rasters based on the number of cells that differ and the average variation of cell values.

In Section 6, we evaluate two distance measures, the Weighted Hamming Distance (WHD) and the Combined Hamming distance (CHD) to analyze the impact of the original Hamming distance on the weighted changes.

5.2. Selection of the number of clusters

Selecting the number of clusters is a crucial aspect of successful clustering. This study employed two strategies to determine a suitable number of clusters for the different clustering techniques.

The first strategy applies the Silhouette index described in Section 2.2. The Silhouette index helps to identify a suitable number of clusters, ensuring better grouping of the temporal clusters. The experimental results in [44] demonstrate that the Silhouette index presents a very good performance. The clustering technique is applied for each possible value of k between

2 and the total number of rasters. Next, the Silhouette index is computed based on the clustering performed. The value of k that produces the highest Silhouette index indicates that k is an appropriate value for the number of clusters.

For comparison purposes, a second straightforward strategy is proposed, which involves selecting the number of snapshots generated by constructing the Heuristic T-k^2-raster. In the example shown in Figure 3a, the number of clusters is 4, representing the number of clusters defined by the T-k^2-raster, as each snapshot represents a separate cluster. This strategy aims to analyze the performance of clustering techniques after using different clusters corresponding to the number of snapshots in their respective T-k^2-raster.

5.3. Selection of snapshots to represent a cluster

The final step involves the selection of the raster snapshot within each cluster. This selection enables the structure to consider the remaining rasters in the cluster as raster logs associated with the snapshot of their respective cluster.

Selecting the ideal snapshot within each cluster is crucial to minimize the final size of the resulting structure. In this study we considered two strategies for selecting a good snapshot within each cluster: First Raster Selection and Best Raster Selection.

In the first strategy, the first raster (in time-order) in each group is selected as the cluster snapshot. The strategy maintains similarities concerning the Heuristic T-k^2-raster heuristic. In both cases, the first raster that the cluster represents is defined as a snapshot and the other rasters, temporarily located in the future, are referenced to a snapshot raster temporarily located in the past.

The second strategy selects the raster within the cluster that reduces the sum of the distances to all the other rasters in the cluster, where the distance value corresponds to the distance definition used in the clustering step.

Figure 3b illustrates the application of clustering on a raster time series using a T-k^2-raster structure. In this scenario, *cs_v* is a relevant structure for identifying the snapshot rasters and establishing the corresponding mappings between logs and their respective snapshot references. It should be noted that, according to the snapshot raster selection, values *s_bv* and *cs_v* would differ, as is the case of the example given. For example, in the case of first raster snapshot selection, the rasters snapshot selected are M_1, M_2 and M_3, while for best raster snapshot selection, rasters snapshot are M_2, M_6 and M_7.

6. EXPERIMENTAL RESULTS AND DISCUSSION

This section presents the experimental results that evaluate the different strategies described in the previous sections. Firstly, we assess how far the space obtained by the Heuristic T-k^2-raster is from an optimal strategy of snapshot selection for that structure obtained through the DP technique described in Section 4. Next, we evaluate the approximation that disregards temporal order and uses clustering to potentially achieve better groupings of similar rasters.

6.1. Experimental framework

Server configuration: All the experiments were run on a dedicated Intel® Xeon® Gold 5320T CPU clocked at 2.30 GHz (40 physical cores) with cache sizes 1.9 MB (L1d), 1.3 MB (L1i), 50 MB (L2), and 60 MB (L3), and 252 GB of RAM. The operating system was Debian 11 with kernel 5.10.0-13-amd64. The C++ code was compiled with gcc version 10.2.1 and the -O3 optimizations. The Python code was executed with version 3.9.2.

Implementation code: The distance measures presented in Section 5.1 were implemented using the C++ programming language. The clustering algorithms were implemented in Python using the popular `scikit-learn` library [45]. For K-means (or K-medoids as used in the library), the initialization method of `k-medoids++` was used to select each cluster's representatives or centroids. The `scikit-learn` library also provides a function for computing the Silhouette index for each applied clustering method [30].

All the code was made available at a public repository[3], including the implementation of the clustering techniques discussed in Section 2.2, and the two snapshot selection strategies discussed in Section 5.3.

Datasets: In this study, we use real world, synthetic, and semi-synthetic datasets[4]. Regarding the real datasets, we used the NLDAS-2 collection. This collection was obtained from [46] and it was also used in the experimental process of [3]. This collection is a product of the North American Land Data Assimilation System (NLDAS). It includes information of precipitation and flows across North America from 1979 up to the present, such as surface temperature, humidity, and radiation, among other variables. These raster time series have an hourly time resolution and a spatial resolution of 1/8 degrees. The specific dataset used in our experiments correspond to the time period from January to December 2018. Table 2 presents detailed information regarding the datasets.

[3] `https://gitlab.com/mmunocan/clustering/`

[4] The data underlying this article will be shared on reasonable request to the corresponding author.

Dataset	Minimum value	Maximum value	Unique values	Average value	Standard deviation
APCP	-1	10258	3268	5.43	38.08
CONVfrac	-1	100	102	1.12	9.87
DLWRF	9919	47911	24167	22147.16	8254.10
PEVAP	-79	185	263	-10.05	40.21
SPFH	-1	2	4	-0.19	0.48

TABLE 2: Main statistics of real world datasets. All datasets presents 224 rows, 464 columns and 2664 rasters. APCP represents accumulated precipitation [mm], CONVfrac represents fraction of total precipitation that is convective, DLWRF represents downward longwave radiation flux [W/m^2], PEVAP represents potential evaporation [kg/m^2], and SPFH represents specific humidity [kg/kg].

We also use synthetic and semi-synthetic datasets to provide more insight on the evaluation of the clustering approach. Specifically, these datasets are used to test the hypothesis commented in Section 5 that indicates that the proposed clustering strategy helps to reduce the space usage in raster time series that exhibit a cyclic structure, i.e. that the same rasters repeat, either fully or partially, over time. For this, we generated cyclic temporal rasters where a small set of rasters are repeated until the total number of expected rasters is achieved. All the datasets generated have 224 rows, 464 cols, and 2664 rasters for analogy with the real world datasets described above. Both the synthetic and the semi-synthetic datasets are generated analogously using the two strategies described below. Hence, the only difference between them lies in that the rasters forming the seed set S in synthetic datasets are artificially generated, whereas in semi-synthetic datasets, they are real rasters. The two strategies that we apply to generate synthetic and semi-synthetic datasets with different characteristics are as follows:

- *Regular cyclic datasets*: Given a seed set S of rasters, a new dataset is generated by selecting the first x rasters of the set, where $x \in \{12, 24, 36, 48\}$ and repeating them (keeping their original order) until the final size of the dataset is achieved. Note that all the cycles inside the temporal raster have the same size. For example, if $S = \{A, B, C, D, E, F, G, \ldots\}$, $x = 3$ and the expected dataset size is 3, the generated dataset would be $\{A, B, C, A, B, C, A, B, C\}$.
- *Irregular cyclic datasets*: Given a seed set S of rasters, a new dataset is generated by iteratively selecting the first up to x rasters of the set, where $x \in \{12, 24, 36, 48\}$. On each iteration, the number of selected rasters is chosen randomly from the range $[1..x]$. The selected rasters are concatenated until the final size of the dataset is achieved. In this case, the cycles inside each temporal raster might not all have the same size. Using the same parameters of the previous example, a possible generated dataset could be $\{A, B, A, B, C, A, A, B, C\}$.

Synthetic datasets are created by initially generating the seed set of rasters S with randomly assigned cell values. The values, ranging from 0 to 100, are randomly distributed across the cells following a uniform distribution. These datasets replicate scenarios devoid of spatial or temporal localities.

Semi-synthetic datasets are formed using the first x rasters extracted from specific real-world datasets. In particular, we selected APCP and CONVfrac datasets. These datasets emulate scenarios characterized by spatial and temporal locality within a cyclic context.

Table 3 presents the characterization of the synthetic (Irregular Cycle) and semi-synthetic (Irregular CONVfrac and Irregular APCP) datasets generated. The table presents the number of cycles, the average cycle size, and the standard deviation of cycle size. We only present the information about irregular cycles because each generated dataset contains cycles of different random sizes, converting basic information such as the number of cycles and the average cycle size into unpredictable values before the data generation. In the case of the regular cycles datasets, due to all cycles inside these datasets being the same size, it is easy to determine, for example, the total number of cycles generated. For example, in a dataset with regular cycles of length 12, cycle12, there are $2664/12 = 222$ cycles since the total number of raster is fixed to 2664.

Table 4 presents different measures that help to characterize the employed datasets. The *Moran Index* column [47] shows the average result of the Moran Index computed for each raster in the dataset. A value near 1 shows a high spatial autocorrelation or spatial locality, a value near 0 indicates a random distribution, and a value near -1 displays a perfect dispersion. This last value was absent in our results because we do not evaluate datasets with perfect dispersion. The *% of variation* column presents the average result of the percentage of variation comparing each raster with its predecessor. More precisely, the percentage of variation is the number of cells that differs comparing two rasters divided by the total number of cells inside a raster. A percentage close to 0 presents a high similitude between contiguous rasters or temporal locality.

The last three columns measure the cyclicity of the

Dataset	Maximal cycle size	Number of cycles	Average cycle size	Standard deviation
Irregular Cycle	12	408	6.52	3.40
Irregular Cycle	24	216	12.32	6.90
Irregular Cycle	36	145	18.21	10.37
Irregular Cycle	48	108	24.48	13.87
Irregular `CONVfrac`	12	415	6.40	3.46
Irregular `CONVfrac`	24	214	12.39	6.81
Irregular `CONVfrac`	36	146	18.24	10.27
Irregular `CONVfrac`	48	109	24.59	13.64
Irregular `APCP`	12	422	6.30	3.46
Irregular `APCP`	24	218	12.15	6.88
Irregular `APCP`	36	138	19.24	10.46
Irregular `APCP`	48	110	23.98	13.82

TABLE 3: Main statistics of synthetic datasets with irregular cycles.

datasets. The *cyclic distance* column shows the average distance between each raster and its most similar raster, namely the raster with the smallest percentage of variation. The most similar raster can be forward or backward in the time serie. The *cyclic % of variation* column shows the average variation between each raster and its most similar raster. This value differs from the percentage of variation because the latter compares each raster with its preceding raster, which is not necessarily the most similar. Finally, the *% of negative distances* column shows the proportion of rasters whose most similar raster is located before it.

Baseline: For our baseline, we used the original implementation of the T-k^2-raster library.[5] This library includes the implementation of all variants of T-k^2-raster and k^2-raster. The implementation of the Heuristic T-k^2-raster, used as the principal baseline, represents each raster using the variant k_H^2-raster described in Section 3.1.

We compared the clustering-based method with other existing methods for representing a raster time series. Specifically, we compared our approach with the original Heuristic T-k^2-raster, as well as with an independent collection of k^2-raster (or k^2-raster Collection) and an independent collection of k_H^2-raster (or k_H^2-raster Collection). This analysis aimed to evaluate the performance and efficiency of our proposed method against these established techniques.

k^2-raster configuration: In order to construct the underlying k^2-raster, it is relevant to select the value of the four parameters, namely k_1, k_2, n_1, and l, described in Section 3.1. In this study, we select four combinations to determine the best parameter values. We decided to extend the configurations to improve the compression of each dataset and to make an effective comparison between the original structures and the clustering application. These four configurations were applied to construct the Heuristic T-k^2-raster, the independent collections, and the T-k^2-raster with clustering. Regarding the application of dynamic programming, we applied the same configuration used for the Heuristic T-k^2-raster.

In order to compare each structure to obtain the best result, it is crucial to prepare the structure using the configuration that generates the smallest representation size of the raster time series. In this study, we chose the configuration that achieves the smallest representation size of the raster time series for each dataset and structure evaluated. We can obtain the most efficient representations by carefully selecting the parameters and configurations for each structure. This selection ensures the comparison of the structures under optimal conditions, enabling accurate and meaningful comparisons.

In the case of the k^2-raster Collection, k_H^2-raster Collection, and Heuristic T-k^2-raster, we chose the best values of k_1, k_2, n_1 and l. For all combinations, the values of k_1 and k_2 were kept constant at 4 and 2, respectively. For n_1 and l, we select the best value between 3 and 4 for n_1 and between 1 and 2 for l. Table 5 shows the selected configurations for each representation and dataset. In some instances, two configurations were chosen because both yield an equal-size representation.

6.2. Evaluating the optimality of the heuristic used by the T-k^2-raster

This section presents the results obtained from a comparative analysis between the heuristic for selecting snapshots in the context of the T-k^2-raster and applying dynamic programming (DP) for the same purpose. This comparison aims to evaluate the effectiveness of the heuristic approach in achieving a more compact representation of the data structure.

Table 6 compares the structure size resulting from

[5] `https://gitlab.lbd.org.es/fsilva/k2-raster`

Dataset	Moran Index	% of variation	Cyclic distance	Cyclic % of variation	% negatives distances
APCP	0.87	10.60	1.03	7.23	49.23
CONVfrac	0.81	1.85	4.19	0.86	51.28
DLWRF	0.97	40.71	1.00	22.39	51.20
PEVAP	0.97	19.56	1.03	0.02	66.64
SPFH	0.96	0.26	3.06	0.23	49.25
cycle12	0.00	99.00	12	0.00	99.55
cycle24	0.00	99.00	24	0.00	99.10
cycle36	0.00	99.00	36	0.00	98.65
cycle48	0.00	99.00	48	0.00	98.20
irre_cycle12	0.00	97.79	8.77	0.00	54.02
irre_cycle24	0.00	98.72	17.55	0.00	51.42
irre_cycle36	0.00	98.89	26.07	0.00	52.78
irre_cycle48	0.00	98.91	34.88	0.00	50.29
CONVfrac12	0.69	0.17	12	0.00	99.55
CONVfrac24	0.66	0.20	24	0.00	99.10
CONVfrac36	0.66	0.15	36	0.00	98.65
CONVfrac48	0.64	0.14	48	0.00	98.20
irre_CONVfrac12	0.72	0.19	8.71	0.00	53.72
irre_CONVfrac24	0.69	0.21	17.65	0.00	51.43
irre_CONVfrac36	0.67	0.19	25.98	0.00	51.77
irre_CONVfrac48	0.66	0.17	34.19	0.00	50.68
APCP12	0.79	4.96	12	0.00	99.55
APCP24	0.76	4.48	24	0.00	99.10
APCP36	0.77	3.84	36	0.00	98.65
APCP48	0.76	4.08	48	0.00	98.20
irre_APCP12	0.80	5.70	8.76	0.00	54.02
irre_APCP24	0.78	5.02	17.32	0.00	51.19
irre_APCP36	0.77	4.44	26.37	0.00	50.55
irre_APCP48	0.77	4.27	35.20	0.00	50.16

TABLE 4: Detailed characterization of the datasets. Moran Index measures the spatial locality, Percentage of variation measures the temporal locality, and the last three columns characterize the cyclicity of the datasets.

Dataset	k^2-raster Collection	k^2_H-raster Collection	Heuristic T-k^2-raster
APCP	4-2-3-1, 4-2-3-2	4-2-3-1	4-2-3-2
CONVfrac	4-2-3-1, 4-2-3-2	4-2-3-1	4-2-4-1
DLWRF	4-2-4-1, 4-2-4-2	4-2-4-2	4-2-4-1
PEVAP	4-2-3-1, 4-2-3-2	4-2-3-1	4-2-3-1
SPFH	4-2-4-1	4-2-4-1	4-2-3-2, 4-2-4-2

TABLE 5: Best configuration chosen for each structured compared. Each configuration is presented as k_1-k_2-n_1-l. For structures with two configurations, we choose the first option for queries evaluation.

Dataset	Heuristic T-k^2-raster	Dynamic Programming	Improvement percentage
APCP	43.74	43.73	0.02
CONVfrac	12.87	12.82	0.39
DLWRF	298.45	298.45	0.00
PEVAP	27.66	27.66	0.00
SPFH	5.35	4.94	7.66

TABLE 6: Comparison of the structure size in MB of the T-k^2-raster applying the heuristic vs DP. The configuration for DP is the same configuration presented in the last column in Table 5. To determine the percentage of improvement, the formula $\frac{bs-dp}{bs} \times 100$ was used, where bs denotes the heuristic T-k^2-raster size and dp represents the size result applying DP.

applying the heuristic[6] and DP strategy. In those results, we applied the same configuration presented in the last column in Table 5. The improvement percentage shows that the structures size are very close to each other for APCP and CONVfrac datasets, while for DLWRF and PEVAP datasets the results have not changes. These results show that the snapshot selection is very close between the heuristic and DP strategies or they are equal for both datasets with the same structure size. Only for the case of SPFH it is possible to observe a more

[6]The sizes reported for the baseline structure differ from the ones in [3] because the original study swapped rows and columns producing larger sizes.

relevant difference than the other datasets, but since this dataset is the smallest, the difference is negligible.

The results indicate that the heuristic employed in the Heuristic T-k^2-raster is efficient, as it can achieve a compact size close to the optimal result obtained through dynamic programming. These findings highlight the effectiveness of the heuristic approach in achieving efficient compression sizes for raster time series data.

6.3. Evaluating the application of clustering to T-k^2-raster

This section presents a comparative analysis of the implementation of the clustering technique versus k^2-raster Collection, k_H^2-raster Collection, and Heuristic T-k^2-raster. The main objective is to compare the effects of clustering against the baseline previously described, particularly the heuristic strategy. The clustering strategy should expand the range of available rasters snapshots options for each raster log, providing better alternatives that contribute to reducing the size of the data structure. In this context, we include the synthetic and semi-synthetic datasets that present a cyclic temporal behavior in this analysis. Those datasets may take advantage of the additional alternatives of snapshot selection that clustering offers.

Parameters tuning: For clustering applications, the first step is to precompute the distance matrix. It corresponds to a square matrix that stores the distance values of all the rasters with each other. With the precomputed distance matrix, the second step is to apply the corresponding clustering technique and, later, the snapshot selection. The results correspond to the cs_v vector that the T-k^2-raster adapted required as an input. Tables 5 and 8 show the configuration used for each data structure.

Table 7 presents the number of clusters selected for this experimental process. The first strategy selection is based on the Silhouette Index result, while the second is based on the number of snapshots defined by the heuristic.

Space evaluation: Table 9 presents the size of the data structures compared with the corresponding percentage of improvement. When comparing the clustering results with k^2-raster Collection and k_H^2-raster Collection, the clustering improvement percentage is between 0.38% and 68.60%. This suggest that clustering techniques for snapshot selection provides better compression results for rasters time series than both k^2-raster and k_H^2-raster collection which do not use snapshots selection.

Conversely, the improvement decreases when comparing the cluster results with the Heuristic T-k^2-raster. `APCP` and `CONVfrac` datasets present a negative improvement where the clustering technique increases the structure size. Table 7 explains the results by a low Silhouette Index. A low Silhouette Index indicates the difficulty of the clustering technique in finding an optimal cluster number.

`SPFH` and `PEVAP` present competitive results, where clustering size reaches a similar value that Heuristic T-k^2-raster. After a detailed datasets revision, we discovered that the reason is different for both datasets. In the case of `SPFH`, the outcome can be attributed to the dataset's low number of distinct values, resulting in minimal variation between values. For `PEVAP`, the snapshot and cluster selections are the same for clustering and the heuristic strategies. In addition, the Silhouette Index for `PEVAP` indicates an high value (close to 1) according to Table 7.

`DLWRF` is the only case in which the clustering technique reduces the space usage compared with the heuristic. The Silhouette Index also presents a value close to 1 (see Table 7). After a manual inspection of the generated data structures, we discovered that although both strategies select the same clusters, inside each cluster they select different snapshots. Specifically, both techniques select groups of three contiguous rasters as clusters, however, while the heuristic selects the first raster in each cluster as a snapshot, clustering always selects the second raster because it uses the Best Raster Selection strategy explained in Section 5.3. The heuristic approach, by design, systematically selects the first raster within each cluster as the snapshot without considering factors such as its distance to other rasters. Conversely, employing clustering with the Best Raster Selection strategy provides increased adaptability. This method considers the distance between the chosen snapshot and the remaining rasters within the cluster. Consequently, it enables the identification of a more suitable alternative snapshot that effectively minimizes the structure size compared to the heuristic method. This difference allows the clustering technique to obtain an advantage with respect to the heuristic.

We can observe some correspondences if we compare the results obtained in Table 9 with the measures presented in Table 4. On the one hand, `APCP`, `CONVfrac`, and `SPFH` present a low difference between the percentage of variation and its cyclic version. Instead, `DLWRF` and `PEVAP` present a significant difference in both percentages of variation. These results indicate that if the raster variation does not significantly differ comparing a raster with its similar raster versus the last raster, T-k^2-raster structure would not obtain an improvement after applying the clustering technique.

Another interesting result is the high percentage of negative distances on `PEVAP` dataset compared with the rest of the real-world datasets. Including the cyclic distance near 1, we can infer that for each raster, the most similar raster is usually located immediately before it. This scenario could be optimal for the heuristic strategy, such as the case of `PEVAP`.

Dataset	Hierarchical WHD	Hierarchical CHD	Kmedoids WHD	Kmedoids CHD	Configuration k_1-k_2-n_1-l			
					4-2-3-1	4-2-3-2	4-2-4-1	4-2-4-2
APCP	2 (0.092)	2 (0.428)	12 (-0.045)	10 (-0.577)	889	888	889	858
CONVfrac	2 (0.407)	4 (0.340)	2 (0.254)	15 (0.294)	740	633	685	462
DLWRF	888 (0.999)	888 (0.999)	888 (0.999)	888 (0.999)	888	888	888	888
PEVAP	889 (0.999)	889 (0.999)	889 (0.999)	889 (0.999)	889	889	889	889
SPFH	2 (0.515)	2 (0.599)	2 (0.560)	2 (0.573)	24	14	14	26

TABLE 7: Selected number of clusters for clustering technique and for the Heuristic. The corresponding Silhouette value is shown in parentheses.

Dataset	Configuration			
APCP	Kmedoids	WHD	Best Centroid	4-2-3-2
CONVfrac	Hierarchical	WHD	Best Centroid	4-2-3-2
CONVfrac	Kmedoids	CHD	Best Centroid	4-2-4-1
DLWRF	Hierarchical/Kmedoids	WHD/CHD	Best Centroid	4-2-4-1
PEVAP	Hierarchical/Kmedoids	WHD/CHD	First/Best Centroid	4-2-3-1
SPFH	Kmedoids	WHD	Best Centroid	4-2-3-2/4-2-4-2

TABLE 8: Best configuration chosen to represent the Heuristic T-k^2-raster based on clustering. The k^2-raster configuration is presented as k_1-k_2-n_1-l. For structures with two or more configurations, we choose the first option for queries evaluation. All configurations use the number of clusters presented in Table 7.

Query time evaluation: To compare query time, we evaluated the three queries implemented in [3]: *Get Cell* (retrieves the value of the cell), *Get values window* (retrieves all the cell values in a rectangular cuboid defined) and *Get cells by values* (retrieves all the cells inside a rectangular cuboid whose values are within a defined range). The results show that the cluster T-k^2-raster has a similar query time performance as the Heuristic T-k^2-raster. Applying the cluster in the T-k^2-raster does not significantly modify query time efficiency. Tables 10a, 10b and 10c present the query time results for the respective queries over real-world datasets.

Cyclic datasets results: Now we focus on the analysis regarding cyclic datasets. To check if the clustering technique can detect the optimal number of clusters, we calculate the Silhouette index for all the synthetic datasets and each possible value of k between 2 and the total number of temporal rasters. The objective was to identify the cluster size that maximized the Silhouette index, which indicates the optimal number of clusters. For all datasets, the computed number of clusters coincides with the cycle length of regular cycle datasets and the maximal cycle length for irregular cycle datasets. These results indicate the effectiveness of the clustering technique in selecting the appropriate number of clusters and the subsequent snapshot selection process.

Table 11 presents the size of the structures and their corresponding percentage of improvement for cyclic datasets. We exclude CONVfrac datasets because it has similar results compared with APCP datasets. Comparing Heuristic T-k^2-raster with the best cluster result, we can observe a significant reduction in the data structure size by applying the clustering technique, especially in the synthetic datasets. Even for datasets with regular cycles, Heuristic T-k^2-raster presents a similar size that k_H^2-raster Collection, indicating that the heuristic selects all rasters as a snapshot, similar to a k_H^2-raster Collection. In the case of semi-synthetic datasets, the difference is less because these datasets present a high spatial locality and a more significant temporal locality than synthetic datasets.

7. CONCLUSIONS AND FUTURE WORK

In this work, we delve into the study of efficient space representation for raster time series. Our first objective was to perform a comprehensive study about the optimality of the heuristic used for selecting snapshots in the T-k^2-raster variants, a compact data structure for succinctly representing raster time series. This structure classifies the rasters in snapshots and logs, in which snapshots serve as references to logs. It is important to recall that only the most recently generated snapshot, following the time-order, can be referenced by a log. The heuristic selects a suitable subset of snapshots in order to reduce the space of the representation. To assess the effectiveness of such an heuristic, we compared it with an optimal selection strategy obtained through the use of a dynamic programming algorithm. Our findings indicate that the Heuristic T-k^2-raster achieves a compression rate close to the optimal attainable by any time-ordered variant of the T-k^2-raster.

Subsequently, we studied the potential enhancements

Dataset	Size [MB]				Percentage of improvement (%)		
	k^2-raster Collection	k^2_H-raster Collection	Heuristic T-k^2-raster	Best cluster result	k^2-raster Collection	k^2_H-raster Collection	Heuristic T-k^2-raster
APCP	55.74	54.95	43.74	54.76	1.79	0.38	-25.15
CONVfrac	19.07	19.67	12.87	16.13	16.20	18.76	-24.16
DLWRF	450.54	365.50	298.45	283.19	37.14	22.52	5.11
PEVAP	88.14	82.12	27.66	27.68	68.60	66.29	-0.07
SPFH	7.14	7.66	5.35	5.38	24.51	29.63	-0.75

TABLE 9: Structures size for each dataset and structures compared. To determine the percentage of improvement, the formula $\frac{bs-cl}{bs} \times 100$ was used, where bs denotes the baseline size of the structure and cl represents the size of the T-k^2-raster with clustering.

Dataset	Heuristic T-k^2-raster	Cluster result
APCP	0.37	0.30
CONVfrac	0.23	0.18
DLWRF	0.78	0.80
PEVAP	0.40	0.41
SPFH	0.13	0.15

(a) **Query:** Get cell (in μs/query, 100x100,000 queries)

Dataset	Heuristic T-k^2-raster	Cluster result
APCP	12.81	13.55
CONVfrac	8.12	7.63
DLWRF	26.18	26.87
PEVAP	18.52	19.23
SPFH	6.15	6.91

(b) **Query:** Get values window (in μs/cell, 100x100 queries)

Dataset	Heuristic T-k^2-raster	Cluster result
APCP	34,320.40	32,370.20
CONVfrac	377.55	356.48
DLWRF	101.42	106.98
PEVAP	78.86	81.49
SPFH	17.70	19.76

(c) **Query:** Get cells by value (in μs/cell, 100x100 queries)

TABLE 10: Query time results for Heuristic T-k^2-raster and Clustering

achievable through non-time-ordered variants, employing clustering techniques. In this approach, we group the rasters in the time series based on similarity using various measures derived from the Hamming distance. Next, for each cluster, one raster is selected as a snapshot, while all the others are encoded as logs with respect to the chosen snapshot. We use an array to identify, for each raster, its corresponding snapshot. Our experimental evaluation demonstrates that clustering can achieve an equivalent or improve compression performance for most of the tested datasets of the Heuristic T-k^2-raster while maintaining query support performance. We identify that clustering performs the best in datasets with rasters repeated in cycles.

As part of our future work, we plan to delve deeper into the application of clustering techniques. Specifically, we intend to investigate the implementation of DBSCAN [48] and explore an alternative approach to characterizing the rasters. This alternative approach involves extracting features from each raster, such as its width, height, the maximum/minimum/average values in the raster, and some indexes such as Moran [47], Geary [49], and Getis [50], and use the cosine distance to compute the similarity between the feature vectors that characterize each raster.

ACKNOWLEDGEMENTS

This work was supported by the Agencia Nacional de Investigación y Desarrollo [21200810 to M.M. and FONDECYT grants 11220545 to J.F. and 1-230755 to G.N.]; the Centre for Biotechnology and Engineering [FB0001 to M.M., C.H. and G.N.]; the Agencia Nacional de Investigación y Desarrollo – Millennium Science Initiative Program [ICN17_002 to M.M., J.F. and G.N.]; and PID2022-141027NB-C21 (EarthDL), TED2021-129245B-C21 (PLAGEMIS), PID2020-114635RB-I00 (EXTRACompact), PDC2021-121239-C31 (FLATCITY-POC) and PDC2021-120917-C21 (SIGTRANS): partially funded by MCIN/AEI/10.13039/501100011033 and "NextGenerationEU"/PRTR, GRC: ED431C 2021/53, partially funded by GAIN/Xunta de Galicia [D.S. and F.S.]. CITIC is funded by the Xunta de Galicia through the collaboration agreement between the Department of Culture, Education, Vocational Training and Universities and the Galician universities for the reinforcement of the research centers of the Galician University System (CIGUS).

REFERENCES

[1] Rigaux, P., Scholl, M., and Voisard, A. (2002) *Spatial databases: with application to GIS*. Morgan Kaufmann, Burlington, Massachusetts, USA.

Dataset	Size [MB]				Percentage of improvement (%)		
	k^2-raster Collection	k_H^2-raster Collection	Heuristic T-k^2-raster	Best cluster result	k^2-raster Collection	k_H^2-raster Collection	Heuristic T-k^2-raster
Cycle12	345.41	260.81	260.81	1.59	99.54	99.39	99.39
Cycle24	345.45	260.83	260.83	2.77	99.20	98.94	98.94
Cycle36	345.43	260.82	260.82	3.94	98.86	98.49	98.49
Cycle48	345.43	260.81	260.81	5.11	98.52	98.04	98.04
Irregular_Cycle12	345.45	260.80	257.61	1.59	99.54	99.39	99.38
Irregular_Cycle24	345.42	260.81	260.09	2.77	99.20	98.94	98.93
Irregular_Cycle36	345.46	260.82	260.43	3.94	98.86	98.49	98.49
Irregular_Cycle48	345.42	260.82	260.58	5.02	98.55	98.08	98.07
APCP12	25.25	27.43	18.41	0.54	97.86	98.03	97.07
APCP24	24.19	26.39	16.92	0.66	97.27	97.50	96.10
APCP36	22.31	24.31	14.89	0.74	96.68	96.96	95.03
APCP48	23.25	25.20	15.31	0.87	96.26	96.55	94.32
Irregular_APCP12	26.18	28.59	18.53	0.54	97.94	98.11	97.09
Irregular_APCP24	25.19	27.44	17.74	0.66	97.38	97.59	96.28
Irregular_APCP36	25.90	26.01	16.50	0.74	97.14	97.15	95.52
Irregular_APCP48	23.44	25.47	15.93	0.87	96.29	96.58	94.54

TABLE 11: Structures size for each dataset and structures compared. To determine the percentage of improvement, the formula $\frac{bs-cl}{bs} \times 100$ was used, where bs denotes the baseline size of the structure and cl represents the size of the T-k^2-raster with clustering.

[2] Worboys, M. F. and Duckham, M. (2004) *GIS: a computing perspective*. CRC press, Boca Raton, Florida.

[3] Silva-Coira, F., Paramá, J. R., de Bernardo, G., and Seco, D. (2021) Space-efficient representations of raster time series. *Information Sciences*, **566**, 300–325.

[4] Bhagat, A. P. and Atique, M. (2012) Medical images: Formats, compression techniques and dicom image retrieval a survey. *Proceedings of 2012 International Conference on Devices, Circuits and Systems (ICDCS)*, Piscataway, New Jersey, United States, 3, pp. 172–176. IEEExplore.

[5] Erickson, B. J., Manduca, A., Palisson, P., Persons, K. R., Earnest 4th, F., Savcenko, V., and Hangiandreou, N. J. (1998) Wavelet compression of medical images. *Radiology*, **206**, 599–607.

[6] Koff, D. A. and Shulman, H. (2006) An overview of digital compression of medical images: can we use lossy image compression in radiology? *Canadian association of radiologists journal*, **57**, 211.

[7] Maireles-González, O., Bartrina-Rapesta, J., Hernández-Cabronero, M., and Serra-Sagrista, J. (2022) Analysis of lossless compressors applied to integer and floating-point astronomical data. *Proceedings of 2022 Data Compression Conference (DCC)*, Piscataway, New Jersey, United States, 3, pp. 389–398. IEEExplore.

[8] Chow, K., Tzarmarias, D. E. O., Hernández-Cabronero, M., Blanes, I., and Serra-Sagristà, J. (2022) Performance improvement on k^2-raster compact data structure for hyperspectral scenes. *IEEE Geoscience and Remote Sensing Letters*, **19**, 1–5.

[9] Chow, K., Tzamarias, D. E. O., Hernández-Cabronero, M., Blanes, I., and Serra-Sagristà, J. (2020) Analysis of variable-length codes for integer encoding in hyperspectral data compression with the k2-raster compact data structure. *Remote Sensing*, **12**.

[10] Chow, K., Tzamarias, D. E. O., Blanes, I., and Serra-Sagristà, J. (2019) Using predictive and differential methods with k2-raster compact data structure for hyperspectral image lossless compression. *Remote Sensing*, **11(21):2461**.

[11] Zhang, J. and You, S. (2013) High-performance quadtree constructions on large-scale geospatial rasters using gpgpu parallel primitives. *International Journal of Geographical Information Science*, **27**, 2207–2226.

[12] Ladra, S., Paramá, J. R., and Silva-Coira, F. (2017) Scalable and queryable compressed storage structure for raster data. *Information Systems*, **72**, 179–204.

[13] Li, Y. and Bretschneider, T. R. (2007) Semantic-sensitive satellite image retrieval. *IEEE Transactions on Geoscience and Remote Sensing*, **45**, 853–860.

[14] Quartulli, M. and G. Olaizola, I. (2013) A review of eo image information mining. *ISPRS Journal of Photogrammetry and Remote Sensing*, **75**, 11–28.

[15] Navarro, G. (2016) *Compact data structures: A practical approach*. Cambridge University Press, Cambridge, England.

[16] de Bernardo, G., Álvarez-García, S., Brisaboa, N. R., Navarro, G., and Pedreira, O. (2013) Compact querieable representations of raster data. *Proceedings of String Processing and Information Retrieval*, 10, pp. 96–108. Springer,Cham.

[17] Brisaboa, N. R., Cerdeira-Pena, A., de Bernardo, G., Navarro, G., and Óscar Pedreira (2020) Extending general compact querieable representations to gis applications. *Information Sciences*, **506**, 196–216.

[18] Pinto, A., Seco, D., and Gutiérrez, G. (2017) Improved queryable representations of rasters. *Proceedings of 2017 Data Compression Conference (DCC)*, Pis-

cataway, New Jersey, United States, 4, pp. 320–329. IEEExplore.

[19] Ladra, S., Paramá, J. R., and Silva-Coira, F. (2016) Compact and queryable representation of raster datasets. *Proceedings of the 28th International Conference on Scientific and Statistical Database Management*, New York, NY, USA, 6. Association for Computing Machinery.

[20] Cerdeira-Pena, A., de Bernardo, G., Fariña, A., Paramá, J. R., and Silva-Coira, F. (2018) Towards a compact representation of temporal rasters. *Proceedings of String Processing and Information Retrieval*, 10, pp. 117–130. Springer,Cham.

[21] Brisaboa, N. R., Ladra, S., and Navarro, G. (2009) k^2-trees for compact web graph representation. *Proceedings of String Processing and Information Retrieval*, Berlin, Heidelberg, 8, pp. 18–30. Springer Berlin Heidelberg.

[22] Brisaboa, N. R., Ladra, S., and Navarro, G. (2014) Compact representation of web graphs with extended functionality. *Information Systems*, **39**, 152–174.

[23] Ladra, S. (2011) Algorithms and compressed data structures for information retrieval. PhD thesis Universidade da Coruña.

[24] Samet, H. (1984) The quadtree and related hierarchical data structures. *ACM Comput. Surv.*, **16**, 187–260.

[25] Anand, R. and Jeffrey David, U. (2011) *Mining of massive datasets*. Cambridge University Press, Cambridge, England.

[26] Rokach, L. and Maimon, O. (2005) Clustering Methods. In Maimon, O. and Rokach, L. (eds.), *Data Mining and Knowledge Discovery Handbook*. Springer US, Boston, MA.

[27] Pena, J. M., Lozano, J. A., and Larranaga, P. (1999) An empirical comparison of four initialization methods for the k-means algorithm. *Pattern recognition letters*, **20**, 1027–1040.

[28] Steinley, D. (2006) K-means clustering: a half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, **59**, 1–34.

[29] Arthur, D. and Vassilvitskii, S. (2007) K-means++ the advantages of careful seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, New York, NY, USA, 1, pp. 1027–1035. Association for Computing Machinery.

[30] Rousseeuw, P. J. (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, **20**, 53–65.

[31] Brisaboa, N. R., Ladra, S., and Navarro, G. (2013) Dacs: Bringing direct access to variable-length codes. *Information Processing & Management*, **49**, 392–404.

[32] Alkathiri, M., Abdul, J., and Potdar, M. B. (2017) Kluster: Application of k-means clustering to multi-dimensional geo-spatial data. *Proceedings of 2017 International Conference on Information, Communication, Instrumentation and Control (ICICIC)*, Piscataway, New Jersey, United States, 8, pp. 1–7. IEEExplore.

[33] Alkathiri, M., Jhummarwala, A., and Potdar, M. (2019) Multi-dimensional geospatial data mining in a distributed environment using mapreduce. *Journal of Big Data*, **6**, 82.

[34] Alzaghoul, E., Al-Zoubi, M. B., Obiedat, R., and Alzaghoul, F. (2021) Applying machine learning to dem raster images. *Technologies*, **9**.

[35] Veda Sai Rochishna, E., Ganesh Hari Prasad Rao, V., Bhargav, G., and Sathyalakshmi, S. (2023) Lossless image compression using machine learning. *Proceedings of Sentiment Analysis and Deep Learning*, Singapore, 1, pp. 113–125. Springer Nature Singapore.

[36] Kiran, R. U. (2021) Discovering knowledge hidden in raster images using rasterminer. *Proceedings of the 2021 Workshop on Intelligent Cross-Data Analysis and Retrieval*, New York, NY, USA, 11 ICDAR '21 1. Association for Computing Machinery.

[37] Mariani, A., Epicoco, I., Cafaro, M., and Pulimeno, M. (2023) Grid-based contraction clustering in a peer-to-peer network. *Proceedings of Machine Learning, Optimization, and Data Science: 8th International Workshop, LOD 2022, Certosa di Pontignano, Italy, September 19–22, 2022, Revised Selected Papers, Part II*, Berlin, Germany, 9, pp. 373–387. Springer.

[38] Aghaee, A., Shamsipour, P., Hood, S., and Haugaard, R. (2021) A convolutional neural network for semi-automated lineament detection and vectorisation of remote sensing data using probabilistic clustering: A method and a challenge. *Computers & Geosciences*, **151**, 104724.

[39] Wu, X. and Zhang, X. (2019) An efficient pixel clustering-based method for mining spatial sequential patterns from serial remote sensing images. *Computers & Geosciences*, **124**, 128–139.

[40] Sisodiya, N., Garg, S., Dube, N., Thakkar, P., Parmar, A., and Sharma, S. (2023) Scalable clustering for eo data using efficient raster representation. *Multimedia Tools and Applications*, **82**, 12303–12319.

[41] Hamming, R. W. (1950) Error detecting and error correcting codes. *The Bell System Technical Journal*, **29**, 147–160.

[42] Merigo, J. M. and Casanovas, M. (2010) Decision making with distance measures and linguistic aggregation operators. *International Journal of Fuzzy Systems*, **12**, 190–198.

[43] Merigó, J. M. (2010) Using the probabilistic weighted average in decision making with distance measures. *Proceedings of the World Congress on Engineering*, Kwun Tong, Kowloon, Hong Kong, 6, pp. 1–4. Imperial College London London International Association of Engineers.

[44] Starczewski, A. and Krzyżak, A. (2015) Performance evaluation of the silhouette index. *Proceedings of Artificial Intelligence and Soft Computing*, 6, pp. 49–58. Springer,Cham.

[45] Kramer, O. (2016) *Machine Learning for Evolution Strategies*. Springer,Cham.

[46] Xia, Y., Mitchell, K., Ek, M., Sheffield, J., Cosgrove, B., Wood, E., Luo, L., Alonge, C., Wei, H., Meng, J., et al. (2009) Nldas primary fosrcing data l4 hourly 0.125× 0.125 degree v002. goddard earth sciences data and information services center (ges disc), greenbelt, md, usa, rep. Technical report. NASA/GSFC/HSL, Maryland, Washington DC, USA.

[47] Moran, P. A. (1950) Notes on continuous stochastic phenomena. *Biometrika*, **37**, 17–23.

[48] Khan, K., Rehman, S. U., Aziz, K., Fong, S., and Sarasvady, S. (2014) Dbscan: Past, present and future. *Proceedings of the fifth international conference on the applications of digital information and web technologies (ICADIWT 2014)*, Piscataway, New Jersey, United States, 2, pp. 232–238. IEEExplore.

[49] Unwin, A. (1996) Geary's contiguity ratio. *The Economic and Social Review*, **27**, 145–159.

[50] Ord, J. K. and Getis, A. (1995) Local spatial autocorrelation statistics: distributional issues and an application. *Geographical analysis*, **27**, 286–306.