# Using Parallel Hierarchical Clustering to Address Spatial Big Data Challenges

Alan Woodley[1], Ling-Xiang Tang[2], Shlomo Geva[3], Richi Nayak[4,] Timothy Chappell[5]
Queensland University of Technology, Brisbane, Australia
and CRC for Spatial Information
a.woodley[1]|l4.tang[2]| s.geva[3]| r.nayak[4]| t.chappell[5]@qut.edu.au

***Abstract*— Clustering can help to make large datasets more manageable by grouping together similar objects. However, most clustering approaches are unable to scale to very large datasets (e.g. more than 10 million objects). The K-Tree is a data structure and clustering algorithm that has proven to be scalable with large streaming datasets. Here, we apply the K-Tree to spatial data (satellite images) and extend from a single threaded to a multicore environment. We show that the K-Tree is able to cluster larger datasets more efficiently than baseline approaches.**

***Keywords— big data, clustering, remote sensing.***

Satellite images have been analyzed to address environmental and social issues for decades [1]. The satellite remote sensing community faces multiple big data challenges due to the volume, velocity and variety associated with image generation. In the future, these challenges will increase, as more satellites become operational and produce higher quality and more frequent images [2, 3].

Clustering groups items together based on similarity. Clustering can be used to aid the analysis of large datasets, for example, as a precursor to data mining [4]. It should be useful to use clustering to aid in the analysis of satellite images, however, the majority of traditional clustering techniques have been established to operate on static datasets, with a small number of objects (typically less than a million) and to cluster them into a small number of groups (typically less than a thousand) [5]. Satellite image analysis is often performed on a pixel-by-pixel basis and a single image often contains millions of pixels. Hence, traditional clustering techniques are unsuitable without relying on heavily sampling [6, 7] which reduces cluster quality [5].

Recently, state-of-the-art clustering techniques have been developed which are able to cluster billions of objects [6, 8-13] - however, they almost always require the use of expensive high performance supercomputers. Furthermore, they often produce a small number of coarse, low quality clusters [6, 8, 9, 11]. They also often employ non-iterative clustering approaches such as MapReduce or Hadoop that require complete re-clustering each time new data is generated [14].

Here, we apply a clustering algorithm and data structure called the K-Tree [15] to satellite images. The K-Tree has specifically been designed for streaming big data and has previously been applied to big data challenges in the text (web) retrieval domain [16-19]. We also extend the K-Tree to a multicore system which required overcoming data integrity obstacles associated with simultaneous updates to memory.

We begin by describing the original K-Tree and the extension in detail. Then, we demonstrate how the K-Tree is able to cluster millions of objects (pixels) three to four orders of magnitude more quickly than baseline approaches. We argue that this demonstrates the potential of the K-Tree to be applied to even large datasets of satellite images.

## I. BACKGROUND

### *A. Remote Sensing and Big Data*

Remote sensing is the monitoring of an object without physical interaction with the object. A common example of remote sensing are satellite images which have been used for socio-environmental analysis for decades [1]. Satellite images are a valuable tool as they provide continuous and affordable global-scale monitoring.

Satellites operate as follows. During the day, sunlight is absorbed or reflected by objects on earth, with different objects reflecting and absorbing wavelengths across the spectrum at different radiometric magnitudes. The radiometric magnitudes are captured by satellites as values across one or more bands, with each band spanning multiple wavelengths. For each class of object, a spectral signature is created via the distribution of values across the bands enabling different classes to be distinguished.

The remote sensing community faces big data challenges, such as:

- Firstly, the *volume* of satellite images will increase by an order of magnitude by 2030 [2];
- Secondly, the *velocity* of satellite data generation is increasing, from a fortnightly or weekly generation to daily [3]; and
- Thirdly, satellite images have a high degree of *variety* since the same area can completely different in images captured on different dates – even if those dates are close together, for example, before, during and after a flood.

Despite the success of remote sensing analysis, most approaches are suitable for small datasets, for example heavily sampling datasets to analyze only one image per location per year [20] and do not scale to larger datasets. Some emerging approaches have analyzed larger datasets [21, 22] but are computationally expensive and do not scale. This shows the need for the remote sensing community to utilize techniques specifically designed to handle big data.

### B. Clustering Big Data

Clustering organizes objects into groups that share similar properties. Clustering is often used a precursor to other data mining and data analysis tasks [23]. The types of clustering approaches include: those which separate objects into different partitions such as k-means [24], k-means++ [25], k-means|| [26]; those with organize clusters into a hierarchy such as BIRCH [27], ROCK [28] and Chameleon [29]; those which group objects with a higher conceptual density such as DBSCAN [30], OPTICS [31], DBCLASD [32] and DENCLUE [33]; those which separate the objects into a number of grids and then cluster within those grids such as Wavecluster [34], STING [35] and OptiGrid [36] and finally those which optimize the fit between the data and a mathematic model such as Expectation Maximization (EM) [37], COWEB [38] and Self Organizing Maps (SOM) [39].

Despite their widespread use, most traditional techniques are unsuitable for big data. For example, k-means [24]: operates with linear complexity to the number of objects and clusters; requires the entire dataset to held in memory perpetually and would need to re-cluster the entire dataset after new data arrives. Hence, it is only suitable for clustering a small number of objects into a small number of clusters (typically less than a million objects into less than a thousand clusters [23]).

More recent research has focused on clustering algorithms specifically designed for big data. However, reviews of a total of 24 [40], 22 [41], 17 [42], 7 [5], 6 [43] and 4 [23] big data clustering algorithms found that they could not scale to large, streaming and highly variable datasets. The most successful approaches have used state-of-the-art parallelization algorithms, such as MapReduce [8, 9, 44, 45] or Mahoot [46]. However, these approaches have only been able to cluster large datasets (that is those with millions – billions of objects) with the use of an expensive supercomputer and are bound to non-iterative parallelization techniques, meaning that they are unsuitable for streaming data as the entire dataset would have to be clustered whenever new data arrived.

The current inability to cluster large datasets is because he underlying data structures, clustering algorithms and parallelization approaches are not designed to handle data that is simultaneously high in volume, velocity and variety – the exact properties of remote sensing data. Here, we present the K-Tree: a data structure and clustering algorithm that is able to: gracefully scale to a large number of objects and clusters; handle streaming data and handle data with high variety by producing fine grained clusters.

### C. K-Tree

The K-Tree is a data structure and clustering algorithm that has been specifically designed for big data. The K-Tree stores objects in a hierarchy. The leaf nodes act as clusters and contain objects, while non-leaf nodes contain representatives (here: means) of the clusters. The K-Tree accepts a single parameter *K* which specifies the number of objects in each leaf node or number of children in each non-leaf node.

The K-Tree is built bottom-up using the following algorithm:

1. The tree is initialized with one empty node.
2. Objects are read from input and added to the node with multidimensional objects represented as vectors.
3. When the cluster becomes full (using the prescribed value of *K*) it is split into two, using an existing algorithm (here: k-means) with vectors grouped into the two new clusters. A third cluster is created to act as a parent and contains the representation of each child. This process is called 'promotion' and is how the tree grows.
4. Objects are continually read from input and inserted into their correct leaf node (cluster) via a nearest-neighbor search from the root, using a predefined distance metric (here: Euclidian distance). After each insertion, representations along the insertion branch are updated.
5. When a node becomes full a split and promotion occur and the tree grows. When the root becomes full a new root is created through a split and promotion and the depth of the tree increases.
6. Insertions continue until all objects have been inserted.

The K-Tree has been able to cluster up to 450 million documents into hundreds of thousands of clusters [16, 17,19, 47] while performing comparably to alternative approaches (Support Vector Machines, k-means and approaches in the CLUTO toolkit).

Overall, the original K-Tree algorithm made progress to addressing big data challenges, namely:

- It addressed the *volume* challenge since it had logarithmic search complexity, whereas other approaches had linear (or worse) search complexity.
- It addressed the *velocity* challenge since the entire dataset did not need to be clustered every time new data was generated, instead relying on the search-insert-split-promote paradigm.
- It addressed the *variety* challenge by building many fine-grained clusters and not requiring dataset sampling.

Despite these strengths, the original K-Tree algorithm also contained limitations since it was a serial algorithm, and so could not take full advantage of multicore systems which are ubiquitous in modern computing. Here, we address the

limitations by exploiting the structure of the K-Tree to enable more efficient clustering in a multicore environment.

## II. MULTICORE K-TREE

The original K-Tree algorithm was a serial implementation since multiple objects could not be inserted into the tree simultaneously. Instead, objects were inserted one-by-one and after each insertion an update was performed to the representations in the relevant branch. This ensured that parent nodes were always accurate representations of their child nodes, but meant that determining the relevant branch was dependent on previous insertions. If multiple objects were inserted simultaneously, then a node could be subjected to multiple simultaneous updates. This would obfuscate the K-Tree algorithm and introduce vexed issues related to simultaneous access of shared memory during implementation.

Previous attempts to address this issue have used a 'delayed update' approach, whereby, an update occurred after $n$ insertions [5]. However, the tradeoff from this approach is that the parents are no longer exact representations of their child nodes. The solution presented here advances on the 'delayed update' approach by allowing multiple objects to be inserted simultaneously, while ensuring that (almost all) parent nodes remain exact representations of their child nodes. Our solution uses the following algorithm:

1. A 'sample' K-Tree is built using a representative subset of the data using the process outlined previously.
2. All nodes in the K-Tree, except for the root node and nodes on the first level (the root node's child nodes) are emptied.
3. Objects are streamed into the K-Tree and assigned to the relevant branch as before, however, each first level node is treated as a separate K-Tree. Updates are made within a branch but first level nodes are not updated.
4. Inserts continue until the dataset is exhausted. The first level nodes are then updated.

Here, parallelization is possible because each branch below the first level is treated as a separate K-Tree. During implementation, this means a worker and isolated memory space are assigned to each branch. Each worker can only insert a single object at a time, however, multiple workers can operate simultaneously, thereby, allowing the K-Tree to be parallelized on a multicore system. If an object is assigned to a worker before the previous update and insert is complete, then it is placed inside a (first in/first out) queue. There is no updating of shared memory until the very end.

Since each branch is updated up to the first level nodes, the vast majority of parent nodes remain an exact representation of their child nodes. This aligns with one of the original motivations of the K-Tree [15] and addresses the major limitation of the previous approach [5]. Only the first level nodes lose some accuracy (since they are not updated until the end). However, we conclude that this is small tradeoff from analysis of the original K-Tree algorithm.

While in the original K-Tree algorithm an entire branch was updated after each insertion, the impact of this update to the non-leaf nodes, hereafter, referred to as the disruption, is greater at the lower levels. So, while the disruption for the last parent will be inversely proportionally to its number of children, up to $1/k$ for a tree of order $k$, the disruption of the root node will be much smaller, approximately $1/k^L$ for a K-Tree of with $L$ levels. The disruption can be generalized down each level of the K-Tree to $1/k^{L-M}$, where $M$ is the level being investigated, with the root node occurring at level 0. Therefore, at the first level the distortion will be very small (almost non-noticeable) and so provides an acceptable tradeoff for the efficiency gains of parallelization.

Care must be taken when establishing the sample K-Tree in steps 1 and 2. There are two basic techniques that can be used for sampling. The first is to choose a sample that is representative of the entire dataset, for example, choose every $n^{th}$ pixel on the $x$ and $y$ axes. The second is to choose a sample that approaches equal distribution of objects within the first level of the K-Tree. Each approach has its advantages and disadvantages, and the most suitable approach will likely be implementation dependent. For example, the first approach will produce a sample K-Tree more similar to a K-Tree produced using the entire dataset and could be used to explore object distribution within the dataset. In contrast, the second approach will result in faster clustering in the parallelization stage, since objects will be distributed equally amongst workers.

When sampling, one must also ensure that sufficient objects are inserted to produce a K-Tree with at least two levels (since the first two levels are kept post sampling). This can be achieved by choose a suitable tree order ($K$) in comparison with the number of objects being sampled.

## III. RESULTS

For clarity, our experiments have been separated into two: first, we compare the original (non-multicore) K-Tree with a set of baselines, and second, we compare the original K-Tree with the multicore K-Tree. This allows us evaluate the applicably of the K-Tree to remote sensing data and to evaluate the benefit of parallelizing the K-Tree across multiple cores. Since the experiments used the same datasets, computational setting and metrics results are comparable, with the second set of experiments a continuation of the first.

### *A. Dataset and Computational Setting*

Our dataset consisted of an image captured by the Landsat 5 satellite sourced from the Geoscience Australia Datacube [2] [48] and presented in Figure 1. The image is of a regional area near Toowoomba, Queensland (27° S, 151°E) captured on 21 January 2011. The image was chosen since it contained heterogeneous land use and therefore, was a suitable representative of a larger area. The image contained 4,000 by 4,000 pixels and was preprocessed with radiometric, atmospheric and bidirectional reflection (BDRF) corrections. In all the experiments, individual pixels were clustered, and hence, correspond to individual objects. All experiments were performed on a standard 'off-the-shelf' server with 72 Intel 2.3GHZ cores and 512 GB of memory.

Fig. 1. Our Dataset Consisting of Diverse Land Types

### B. Metrics

Our experiments were evaluated across scalability and cluster quality. Scalability was evaluated by recording the execution time with respect to an increasing number of objects and clusters. Cluster quality was recorded by calculating the normalized root mean square error (NRMSE). First, a cluster RMSE was calculated as the square root of the average squared Euclidean distance between each object and its cluster mean. Next, a global RMSE was calculated as the square root of the average squared Euclidean distance between each object and the global mean. Finally, the NRMSE was calculated by dividing the cluster RMSE by the global RMSE. A lower NRMSE indicated higher cluster quality since objects in the cluster were more similar to each other.

### C. Orginal K-Tree versus Baseline Systems

Two baselines were compared against the original K-Tree: k-means [24] and parallel k-means [49, 50]. These baselines were chosen due to their popularity, simplicity and efficiency [23]. The parallel k-means experiment was executed using 72 threads while the other experiments were executed using a single thread. For each experiment the k-means and parallel k-means were executed once per dataset with a maximum number of iterations set to 100.

To evaluate scalability with respect to an increasing number of objects two datasets consisting of 1 million and 16 million pixels were used as input. These datasets consisted of the first 1 million pixels and the complete set of pixels in our input image.

To evaluate the techniques' scalability with respect to an increasing number of clusters the techniques were executed with parameters that produced close to $10^n$ clusters, where $n$ was set to a value between 1 and 5. The comparison was complicated by the fact that while it is possible to define the number of clusters produced by k-means/parallel k-means, the same is not true of the K-Tree. Therefore, the K-Tree was executed multiple times with an order ($K$) set between 10 and 100 (for 1 million objects experiment) and 10 and 200 (for 16 million objects experiment). For each execution, the number of clusters produced on each level of the K-Tree was recorded. The closest number of clusters to $10^n$ were chosen as the input number of clusters for the k-means/ parallel k-means experiments enabling a valid comparison. For the 1 million objects dataset a K-Tree with order 15 was used in all cases (so the execution time is the same), while for the 16 million objects dataset K-Trees with listed orders were used. Our results are presented in Tables I to IV.

TABLE I. EXECUTION TIME FOR 1 MILLION OBJECTS

| Number of Clusters | K-Tree Order | Execution Time (sec) | | |
|---|---|---|---|---|
| | | *K-means* | *Parallel K-means* | *K-Tree* |
| 10 | 15 | 3.709 | 4.686 | 5.144 |
| 106 | 15 | 23.58 | 12.44 | 5.144 |
| 1,044 | 15 | 240.2 | 104.06 | 5.144 |
| 10,292 | 15 | 2,334 | 181.38 | 5.144 |
| 102,802 | 15 | 27,610 | 334.23 | 5.144 |

TABLE II. NRMSE FOR 1 MILLION OBJECTS

| Number of Clusters | K-Tree Order | Normalized Root Mean Square Error | | |
|---|---|---|---|---|
| | | *K-means* | *Parallel K-means* | *K-Tree* |
| 10 | 15 | 0.4182 | 0.4072 | 0.5023 |
| 106 | 15 | 0.1823 | 0.1682 | 0.2436 |
| 1,044 | 15 | 0.1012 | 0.0950 | 0.1341 |
| 10,292 | 15 | 0.0626 | 0.0618 | 0.0827 |
| 102,802 | 15 | 0.0368 | 0.0396 | 0.0496 |

TABLE III. EXECUTION TIME FOR 16 MILLION OBJECTS

| Number of Clusters | K-Tree Order | Execution Time (sec) | | |
|---|---|---|---|---|
| | | *K-Means* | *Parallel K-means* | *K-Tree* |
| 10 | 54 | 57.49 | 52.28 | 204.5 |
| 103 | 11 | 385.8 | 106.21 | 120.4 |
| 1,002 | 183 | 3,747 | 1,397 | 359.8 |
| 10,148 | 57 | 38,920 | 13,560 | 224.4 |
| 100,622 | 19 | 353,200 | 87,300 | 168.1 |

TABLE IV. NRMSE FOR 16 MILLION OBJECTS

| Number of Clusters | K-Tree Order | Normalized Root Mean Square Error | | |
|---|---|---|---|---|
| | | *K-Means* | *Parallel K-means* | *K-Tree* |
| 10 | 54 | 0.4449 | 0.4295 | 0.4795 |
| 103 | 11 | 0.2099 | 0.1928 | 0.2760 |
| 1,002 | 183 | 0.1189 | 0.1089 | 0.1500 |
| 10,148 | 57 | 0.0743 | 0.0702 | 0.0922 |
| 100,622 | 19 | 0.0489 | 0.0510 | 0.0657 |

The results verified a number of hypothesis made throughout this paper. Firstly, as the number of clusters increased the NRMSE decreased, verifying the advantage of fine grained over coarse clustering.

Secondly, in terms of efficiency, the K-Tree outperformed alternatives when a large number of clusters were produced. This is due to the speed of the K-Tree logarithmic search complexity and is a strength of K-Tree's underlying design. In particular, the K-Tree's performance against parallel k-means shows that is not sufficient to simply parallelize a traditional clustering approach.

Finally, the NRMSE of k-means/parallel k-means experiments outperformed the K-Tree in all cases, since k-means/parallel k-means optimized clusters globally while the K-Tree optimized clusters locally. However, this tradeoff is acceptable, given the efficiency of the K-Tree. Furthermore, most of the K-Tree's NRMSE values were taken from non-leaf nodes. The K-Tree leaf nodes, which stored the clusters, all had a lower NRMSE and produced significantly higher number of clusters than their k-means/parallel k-means equivalents. This information is presented in Table V.

TABLE V. LEAF CLUSTER DETAILS FOR TOOWOOMBA DATASET

| Experiment | | K-Tree Details | |
|---|---|---|---|
| **Millions of Objects** | **Number of Clusters** | ***Leaf node NRMSE*** | ***Number of Leaf Clusters*** |
| 1 | 10-102,802 | 0.0496 | 102,802 |
| 16 | 10 | 0.0466 | 438,059 |
| 16 | 106 | 0.0374 | 2,248,826 |
| 16 | 1,044 | 0.0562 | 126,175 |
| 16 | 10,292 | 0.0473 | 407,064 |
| 16 | 102,802 | 0.0402 | 1,291,077 |

### *D. Multicore K-Tree versus the Orginal K-Tree*

To test the multicore K-Tree we clustered the full dataset with a varying number of workers from 1 to 79. K-Trees were built with an order of 85 since this created the largest number of first level nodes (where workers were assigned) in the previous experiment.

Results of the experiment are presented in Figure 2 and Table VI. Figure 2 shows that as the number of workers increased the execution time of the mulitcore K-Tree initially decreased rapidly before reaching a plateau. Table VI shows that this increase does not come at the expense of cluster quality, since the normalized RMSE value remains almost constant regardless of the number of workers. This is likely due to the fine-grained clustering approach of the multicore K-Tree.

These results show the efficiency advantages of multicore K-Tree compared with the original K-Tree, while maintaining the advantages of the K-Tree over baseline approaches, such as the ability to handle streaming data, in a multicore environment.

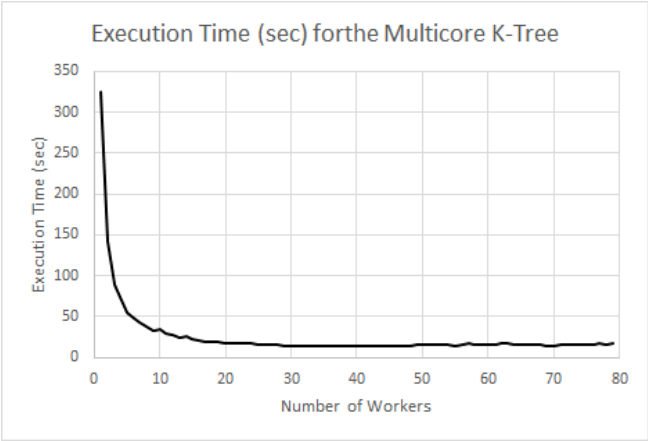


Fig. 2. Multicore K-Tree Execution Time wrt Number of Workers

TABLE VI. RESULTS FROM THE MULTICORE K-TREE EXPERIMENT

| Number of Workers | K-Tree Details | |
|---|---|---|
| | ***Execution Time (sec)*** | ***Normalized RMSE*** |
| 1 | 352.20 | 0.0477 |
| 5 | 55.53 | 0.0482 |
| 10 | 33.51 | 0.0482 |
| 20 | 17.90 | 0.0483 |
| 30 | 14.47 | 0.0476 |
| 50 | 15.91 | 0.0476 |
| 79 | 16.64 | 0.0478 |

## IV. CONCLUSION

Here, we have applied the K-Tree to spatial dataset, and then extended it to a multicore system. We have shown that due to the K-Tree's search operation possessing logarithmic complexity, it is three to five orders of magnitude more efficient than baseline approaches. The multicore extension also increases efficiency up to another order of magnitude. These results show the potential of the K-Tree to be applied to very large, streaming spatial datasets.

## ACKNOWLEDGMENT

This work has been supported by the Cooperative Research Centre for Spatial Information, whose activities are funded by the Business Cooperative Research Centres Programme.

## REFERENCES

[1] W. B. Cohen and S. N. Goward, "Landsat's Role in Ecological Applications of Remote Sensing," *BioScience,* vol. 54, pp. 535-545, June 1, 2004 2004.

[2] M. B. Purss, A. Lewis, A. Ip, and B. Evans, "The new world of 'Big Data' analytics and high performance data: A paradigm shift in the way we interact with very large Earth observation datasets (Invited)," presented at the AGU Fall Meeting Abstracts, San Francisco, 2013.

[3] C. R. Boshuizen, J. Mason, P. Klupar, and S. Spanhake, "Results from the planet labs flock constellation," presented at the 28th Annual AIAA/USU Conference on Small Satellites, Logan, Utah, United States, 2014.

[4] C. C. Aggrawal and C. K. Reddy, *Data Clustering Algorithms and Applications*. New York, United States: CRC Press, 2014.

[5] C. M. De Vries, L. De Vine, R. Nayak, and S. Geva, "Parallel streaming signature EM-tree: A clustering algorithm for web scale applications," presented at the World Wide Web Conference, Florence, Italy, 2015.

[6] Z. Lv, Y. Hu, H. Zhong, J. Wu, B. Li, and H. Zhao, "Parallel K-means clustering of remote sensing images based on mapreduce," presented at the Proceedings of the 2010 international conference on Web information systems and mining, Sanya, China, 2010.

[7] Z. Sun, F. Chen, M. Chi, and Y. Zhu, "A spark-based big data platform for massive remote sensing data processing," presented at the Proceedings of the Second International Conference on Data Science, Sydney, Australia, 2015.

[8] B. Welton, E. Samanas, and B. P. Miller, "Mr. Scan: Extreme scale density-based clustering using a tree-based network of GPGPU nodes," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Los Alamitos, United States, 2013, p. 84.

[9] X. Cui, P. Zhu, X. Yang, K. Li, and C. Ji, "Optimized Big Data K-Means Clustering Using MapReduce," *The Journal of Supercomputing,* vol. 70, pp. 1249–1259, 2014.

[10] X.-J. Wang, L. Zhang, and C. Liu, "Duplicate discovery on 2 billion internet images.," presented at the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Portland, United States, 2013.

[11] Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng*, et al.*, "MR-DBSCAN: An efficient parallel density-based clustering algorithm using MapReduce," in *17th International Conference on Parallel and Distributed Systems (ICPADS)*, Tainan, Taiwan, 2011, pp. 473-480.

[12] T. Liu, C. Rosenberg, and H. A. Rowley, "Clustering billions of images with large scale nearest neighbor search," presented at the IEEE Workshop on Applications of Computer Vision, 2007, Austin, United States, 2007.

[13] B. L. Markham, "Landsat sensor performance: History and current status," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 42, pp. 2691-2694 2004.

[14] A. Mohebi, S. Aghabozorgi, T. Y. Wah, T. Herawan, and R. Yahyapour, "Iterative Big Data Clustering Algorithms: A Review," *Software: Practice and Experience,* vol. 46, pp. 107–129, 2015.

[15] S. Geva, "K-Tree: A height balanced tree structured vector quantizer.," presented at the Workshop on Neural Networks for Signal Processing Sydney, Australia, 2000.

[16] C. M. De Vries and S. Geva, "K-Tree: Large scale document clustering," presented at the 32nd international ACM SIGIR Conference on Research and Development in Information Retrieval, Boston, United States of America, 2009.

[17] C. M. De Vries, L. De Vine, and S. Geva, "Random Indexing K-tree," presented at the Proceedings of the Fourteenth Australasian Document Computing Symposium, School of Information Technologies, Sydney, 2009.

[18] C. M. De Vries, S. Geva, and L. De Vine, "Clustering with random indexing K-tree and XML structure," presented at the INEX 2009, Focused Retrieval and Evaluation : Proceedings of 8th International Workshop of the Initiative for the Evaluation of XML Retrieval, Brisbane, Australia, 2010.

[19] R. Nayak, R. Mills, C. De Vries, and S. Geva, "Clustering and labeling a web scale document collection using Wikipedia clusters," in *Proceedings of the 5th International Workshop on Web-scale Knowledge Representation Retrieval and Reasoning*, Shanghai, 2014, pp. 23-30.

[20] G. Wedderburn-Bisshop, J. Walls, U. Senarath, and A. Stewart, "Methodology for mapping change in woody landcover over Queensland from 1999 to 2001 using Landsat ETM+," presented at the The 11th Australasian Remote Sensing and Photogrammetry Association Conference, Brisbane, Australia, 2002.

[21] Z. Zhu, C. Woodcock, and P. Olofsson, "Continuous monitoring of forest disturbance using all available Landsat imagery," *Remote Sensing of Environment,* vol. 122, pp. 75-91, 2012.

[22] Z. Zhu and C. Woodcock, "Continuous change detection and classification of land cover using all available Landsat data," vol. 144, pp. 152–171, 2014.

[23] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters,* vol. 31, pp. 651-666, 2010.

[24] J. MacQueen, "Some methods for classification and analysis of multivariate observations," presented at the Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley , United States, 1967.

[25] A. D. Vassilvitskii, "K-means++: The advantages of careful seeding," presented at the Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, United States, 2007.

[26] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *Proceedings of the VLDB Endowment,* vol. 5, pp. 622-633, 2012.

[27] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for very large databases," *ACM SIGMOD Record,* vol. 25, pp. 103–114, 1996.

[28] S. Guha, R. Rastogi, and K. Shim, "ROCK: A robust clustering algorithm for categorical attributes," *Information Systems,* vol. 25, pp. 345–366, 2000.

[29] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modelling," *IEEE Computer,* vol. 32, pp. 68–75, 1999.

[30] M. Ester, H.-P. Kriegel, J. Sander, and X. X., "A density-based algorithm for discovering clusters in large spatial databases with noise," presented at the Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining Portland, United States, 1996.

[31] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," *ACM SIGMOD Record,* vol. 28, pp. 49–60, 1999.

[32] X. Xu, M. Ester, H.-P. Kriegel, and S. J., "A distribution-based clustering algorithm for mining in large spatial databases," presented at the Proceedings of the 14th IEEE International Conference on Data Engineering, Washington, United States, 1998.

[33] A. Hinneburg and K. D. A., "An efficient approach to clustering in large multimedia databases with noise," presented at the Proceedings of the ACM SIGKDD Conference on Knowledge Discovery ad Data Mining New York, United States, 1998.

[34] G. G. Sheikholeslami, S. Chatterjee, and A. Zhang, "Wavecluster: A multi-resolution clustering approach for very large spatial databases," presented at the Proceedings of the International Conference Very Large Databases, New York City, United States, 1998.

[35] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining.," presented at the Proceedings of the International Conference Very Large Databases, Athens, Greece, 1997.

[36] A. Hinneburg and D. A. Keim, "Optimal grid-clustering: Towardsbreaking the curse of dimensionality in high-dimensional clustering," presented at the Proceedings of 25th International Conference on Very Large Datanases, Edinburgh, Scotland, 1999.

[37] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society,* vol. 39, pp. 1-38, 1977.

[38] D. H. Fisher, "Knowledge acquisition via incremental conceptual clustering," *Machine Learning,* vol. 2, pp. 139–172, 1987.

[39] T. Kohonen, "The self-organizing map," *Neurocomputing,* vol. 21, pp. 1–6, 1998.

[40] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Zomaya*, et al.*, "A Survey of Clustering Algorithms for Big Data: Taxonomy & Empirical Analysis," *IEEE Transactions on Emerging Topics in Computing,* vol. 2, 2014.

[41] H. Tong, "Big Data Classification," in *Data Classification: Algorithms and Applications*, C. C. Aggarwal, Ed., ed New York, United States of America: CRC Publishing, 2014, pp. 275-283.

[42] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan, "Big data clustering: A review," *Computational Science and Its Applications – ICCSA 2014, Lecture Notes in Computer Science,* vol. 8583, pp. 707-720, 2014.

[43] O. Kurasova, V. Marcinkevicius, V. Medvedev, A. Rapecka, and P. Stefanovic, "Strategies for Big Data Clustering," presented at the IEEE 26th International Conference on Tools with Artificial Intelligence, Limassol, Cyprus, 2014.

[44] A. Kumar, M. Kiran, and B. R. Prathap, "Verification and validation of MapReduce program model for parallel K-means algorithm on Hadoop cluster," *International Journal of Computer Applications,* vol. 72, 2013.

[45] R. Jin, C. Kou, R. Liu, and Y. Li, "Efficient parallel spectral clustering algorithm design for large data sets under cloud computing environment," *Journal of Cloud Computing: Advances, Systems and Applications,* vol. 2, 2013.

[46] R. M. Esteves and C. Rong, "Using Mahout for clustering Wikipedia's latest articles: A comparison between K-means and Fuzzy c-means in the cloud," presented at the IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom), Athens, Greece, 2011.

[47] C. De Vries, "Document clustering Algorithms, Representations and Evaluation for Information Retrieval," Doctor Of Philosophy, Science and Engineering Faculty, Queensland University of Technology, 2014.

[48] N. Mueller, A. Lewis, D. Roberts, S. Ring, R. Melrosea, J. Sixsmith*, et al.*, "Water observations from space: Mapping surface water from 25 years of Landsat imagery across Australia," *Remote Sensing of Environment,* vol. 174, pp. 341–352, 2016.

[49] W.-k. Liao. (2013, 4 August). *Parallel K-Means Data Clustering*. Available: http://users.eecs.northwestern.edu/~wkliao/Kmeans/

[50] H. Bisgin and H. N. Dalfes, "Parallel clustering algorithms with application to climatology," *Geophysical Research Abstracts,* vol. 10, 2008.