2023 3rd International Conference on Computing and Information Technology (ICCIT),
FCIT, University of Tabuk, KSA. 13 and 14/Sep/2023.

Improved Parallel *k*-means Clustering Algorithm

Lama AlGhamdi, Mariam Alkharraa, Sahar AlZahrani
Department of Computer Engineering,
College of Computer Science & Information Technology,
Imam Abdulrahman Bin Faisal University,
Dammam, 31441, Saudi Arabia
{2190002418, 2190005668, 2190003270}@iau.edu.sa

Hajar Bawazir, Wadha AlHajri, Asma AlHajri
Department of Computer Engineering,
College of Computer Science & Information Technology,
Imam Abdulrahman Bin Faisal University,
Dammam, 31441, Saudi Arabia,
{2190009128, 2190004335, 2190001399}@iau.edu.sa

Naya Nagy
Department of Computer Science, Applied College,
Imam Abdulrahman Bin Faisal University,
Dammam, 31441, Saudi Arabia
nmnagy@iau.edu.sa

Mohammed Gollapalli
Department of Computer Information Systems,
College of Computer Science and Information Technology,
Imam Abdulrahman Bin Faisal University,
Dammam, Saudi Arabia
magollapalli@iau.edu.sa

2023 3rd International Conference on Computing and Information Technology (ICCIT) | 979-8-3503-2148-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCIT58132.2023.10273969

***Abstract*— The K-means algorithm, one of the most well-known clustering techniques, has been widely employed to solve a variety of problems. In contrast, the k-means clustering algorithm has numerous restrictions. For instance, the difficulty of dealing with voluminous data, the sensitivity of the outlier, and the random selection of the initial centroid. In this paper, a parallel K-means clustering algorithm is proposed that improves the performance of sequential K-means clustering algorithms by removing outliers from the data before clustering, dividing the data into smaller sections among the threads, and selecting the initial centroid with care. Our primary parallelization tool was OpenMP, which was implemented using the C programming language on 234,296 records. This experiment was conducted using sequential and parallel source code, with modifications made to enhance the parallel functionality. The improved parallel execution resulted in a significant reduction in execution time relative to sequential algorithms. The proposed algorithm source code is also available on GitHub for the community.**

Keywords—k-means clustering, parallel programming, artificial intelligence, OpenMP, machine learning, data science.

I. Introduction

K-Mean is one of the most popular and fundamental unsupervised learning algorithms. Without class labels, unsupervised learning techniques use input vectors to draw conclusions from datasets [1]. Simple K-Mean first determines the number of clusters, K, defines the centroids randomly for data processing, and then assigns each data point to its closest centroid based on the Euclidean distance. The Euclidean distance can occasionally be inaccurate, however, due to the fact that varying relative scaling can lead to distinct clustering [2]. This paper presents a method for determining the distance measure by calculating q and then choosing between the Euclidean and Manhattan distances. Then, the method for selecting initial centroids consistent with the data distribution is provided. The space and processing time requirements of Simple K-Mean clustering methods present the most significant challenges. The Parallel K-Mean clustering algorithm is used to address processing performance and memory requirements for massive datasets. A parallel K-Mean clustering algorithm divides the dataset into smaller sections, computing the smaller sections with less storage and processing power [3].

The disadvantage of the Simple K-Mean algorithm is that it randomly selects the initial centroids. The random selection of the initial centroids causes the results of the Parallel K-Mean clustering technique to vary from run to run. The K-mean clustering method requires the selection of initial centroids with care. Due to the random selection of initial centroids, each run's outcome will differ from the previous one. This flaw causes clusters to differ from one another, and data items within clusters may also differ, affecting the cluster's quality [4]. The improved parallel K-means clustering selects the initial centroids by dividing the data equally into each cluster, calculating the mean of each cluster, and then assigning each data point to its closest centroid. This will ensure that each cluster run is identical, thereby improving cluster quality.

II. Literature Review

In this section, we discuss some previous findings regarding the utilized K-means clustering algorithm, its parallelization, and the techniques and tools that enhance the parallel performance of the K-means clustering algorithm [5]. Since this is a new approach, we were able to highlight the literature review on two of the most influential papers in the field, whose K-means parallelism approaches and algorithms have been extremely beneficial to our research.

This theoretical paper [6] focuses on providing solutions using MapReduce. It proposes using the distance measurement method and the initial selection of centroids to improve the parallel K-means algorithm, which has clear advantages when applied to large data sets. Variable relative scaling can result in diverse clustering, which can cause the Euclidean distance to occasionally lose precision. In addition, outliers result in clusters of subpar data that lose precision and consistency. The strategy is utilized in the modified MapReduce-based parallel K-means algorithm to improve precision. First, calculate q, then select our parallel K-means algorithm's distance measurement strategy. Use the Manhattan distance if q is less than or equal to 8.4595; if q is less than or equal to 8.4595, use the Euclidean distance; and if the data do not support the previous method, use the Euclidean distance.

$$q = \{\frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^4\} / \sigma^4 \qquad (1)$$

There are high-density and low-density portions of the data. Before estimating the size of the object's density area, the radius must be determined. The initial centroids are the most distantly located data objects in the high-density region. The neighborhood of the object contains more than the minimum number of objects, indicating a high population density. Objects with a low density are considered outliers and are removed from the dataset. The rule of distance measure has an average accuracy of 94.73, which is greater than the accuracy of the traditional K-means algorithm and suggests that this method is more widely accepted. Utilizing the initial centroid selection technique reduces the average number of iterations significantly, resulting in a 20% decrease in processing costs. By removing the outliers prior to selecting the initial centroids, clustering results can be made more precise and consistent. In the Distance Measurement Strategy, it is not specified whether to use the Euclidean or Manhattan distance if q=8.4595, or if it is irrelevant.

Using education sector datasets, Shang et al. [7] examined the analysis of simple K-Mean and parallel K-Mean clustering for software products and organizational performance. This study implements and evaluates both Simple and Parallel K-Mean algorithms in order to differentiate their performance characteristics. 10,000 and 5,000 integer data items are used to test and compare the two algorithms. A simple K-Mean clustering algorithm relies on the initial centroids chosen. When different data items are selected in different runs of the same dataset, each run yields distinct results. To circumvent this problem, they utilized a parallel approach, which involved applying the k means algorithm in a parallel context by dividing the dataset into multiple sub-datasets to be distributed across all client systems, where the number of sub-datasets divided is dependent on the number of available client systems connected to the server.

In addition, the experiments conducted by Shang et al. [7] on the algorithm demonstrated significant improvements over its simple equivalent. For instance, the number of iterations in the parallel algorithm is significantly less than the number of iterations in the simple algorithm, and the number of iterations is fixed for each execution/run. In addition, the execution time of the parallel algorithm is significantly less than that of the simple clustering k-means algorithm. According to the authors' comments, additional enhancements can be added to this algorithm to provide a more effective solution. The initial suggestion was to modify the algorithm so that it can be applied to a variety of data types, such as making it applicable to categorical data. Shang et al. also suggested exploring alternative methods for selecting the number of clusters 'k' to apply the K-mean clustering algorithm and incorporating the ability for users to select a 'k' value via parallel settings. In addition, it was suggested that the parallel K-means algorithm be enhanced so that it is applicable to English texts, as it has only been applied to integers thus far. In addition, the Shang et al. procedure contains multiple flaws that have been identified. First, neither the type of parallelism to be applied to this algorithm nor the type of parallel computer the algorithm was built on were specified in the research paper. In addition, it is unclear what type of hardware the tests were conducted on.

After reviewing the literature review, the following gaps were identified. The parallel K-means algorithm is enhanced to be applicable to English texts. As of right now, this algorithm has only been performed on integers. The majority of papers apply each technique individually, as is evident from their respective result tables. Our research overcomes and examines whether parallelization can be improved, as well as whether applying multiple techniques concurrently in parallel K-Means clustering yields superior results to applying each technique individually. Our research also focuses on two methods. Firstly, the process of calculating the initial centroids using the mean, as shown in [7], as opposed to randomly selecting the initial centroids. Secondly, the procedure for calculating the distance denoted by [6] involves calculating q and deciding whether to use Euclidean distance or Manhattan distance.

III. Materials and Methods

In this section, the research and implementation procedure for the simple K-means clustering algorithm is demonstrated, along with its parallel variation, and the improved parallel algorithm is introduced. Moreover, the tools for the experimental process, such as the programming languages, environments and IDEs used for the experimentation, will be discussed in this section. Lastly, the section discusses the pre-processing steps for the dataset used for testing and implementation.

A. Experimental Data

The dataset chosen for the experimentation and testing of the proposed algorithm was taken from data.gov, which is a governmental website that contains real datasets hosted by the by the U.S. General Services Administration. The dataset is created in 2018 and refers to the cook county employees' payroll [8]. The experimentations were conducted on the class attribute of the dataset, which consists of the salaries for all the employees on the records. After conducting the preprocessing techniques on the dataset, which includes removing negative values in the salary attribute, removal of null values, etc., the size of the dataset experimented in this research included 234,296 records.

B. Experimental Setup

Many tools were utilized for the implementation and testing procedure of the K-means algorithms [9]. For programming languages, C was the main choice for the implementation and testing of the simple as well as the parallel variation of the K-means clustering algorithm. Moreover, python is used in this paper for the preprocessing and preparation of the US salaries dataset. Pandas' library is heavily used in data cleaning and preprocessing contexts. Hence, it was chosen as the main tool for dataset preparation and preprocessing. For algorithm parallelization, we used OpenMP as our main parallelization tool, which was incorporated using C language., as it is excellent for utilizing the CPU and multi-cores for conducting multi-threading. Moreover, it contains features that allow for more flexibility in parallel execution, such as the manipulation of thread distribution procedure. This experiment was applied using a laptop that has Intel(R) core(TM), I7-8550U @ 1.80GHz, 1992 Mhz, 4 Cores , 8 logical processors.

C. Simple K-Means Clustering Algorithm

The simple K-means clustering follows a straightforward approach of points distribution, centroid initialization, and clustering procedure [10]. The number of clusters is explicitly selected which is demonstrated by the value K selected before running the program. When running the program centroid values will be randomly initiated for each cluster, the clusters at this point don't have any datapoints assigned to them. upon initiating each centroid, it will initiate each point with a random value. After that it will calculate the distance between the centroids and datapoints, which then will distribute the data points to different clusters according to the shortest centroid distance. After distributing the datapoints, multiple other iterations will be conducted repeating the process and shifting the datapoints into the appropriate clusters until all datapoints are assigned in the correct set of clusters or the maximum iterations is reached. The source code of the simple K-means algorithm was obtained from GitHub, which was written in c language [11].

D. Parallel K-Means Clustering Algorithm

One of the main problems of the simple K-means clustering algorithm is its long execution duration. Hence, a parallel variation of the K-means algorithm has been introduced to produce a shorter execution time for the K-means algorithm by utilizing all the threads found in CPU simultaneously. In parallel K-means clustering, the process of points initialization and clustering is distributed into two separate threads that perform synchronously. Many approaches for K-means clustering parallelization has been introduced. However, in this variation, we have implemented the following parallelization changes into simple K-means:

- Parallelizing the process of initializing the points and clusters by distributing it into two separate sections, where each section is taken by one of the initialized threads.
- Parallelization of the distance calculation using Euclidean and assigning datapoints to clusters by dividing it simultaneously throughout all active threads using the 'static' parameter in OpenMP.
- Resolving possible race condition in the adding the points to clusters process by using the 'atomic' feature in OpenMP.

The discussed parallel K-means clustering algorithm was based on the same GitHub source of the simple K-means [12].

E. Improved Parallel K-Means Clustering Algorithm

The original code uses random values in each execution run, hence the quality of the clusters effects and can't be measured and compared [13-15]. For this reason, we have adjusted the original code to read the data points from a file rather than randomly created.

The parallel K-means previously discussed contained the following limitations which hindered the parallelization performance: a. The clusters centroids are randomly generated and change in every execution. Hence, in every execution, the number of points in each cluster changes significantly. b. The selected distance function is the Euclidean distance. However, because varied relative scaling can result in different grouping, the Euclidean distance can occasionally become inaccurate.
An improved variation of the algorithm is designed in aims to manage the disadvantages found in the previous parallel algorithm [16]. Our aim is to test the possibility of enhancing the k-means algorithm itself by applying more than one technique simultaneously well as improving the parallelization of the algorithm. Our proposed enhancement applied in the parallel K-means clustering algorithm is as follows:

Initial Centroids Calculation: The process of calculation of the initial centroids using the mean as represented in [7] rather than selecting the initial centroids randomly.

- The number of datapoints in each cluster is decided by dividing the number of datapoints with the K value chosen [17].
- The initial centroids are decided based on the mean value of datapoints located in each cluster. The mean is calculated by dividing the number of datapoints for a single cluster with the sum of all clusters' datapoints.

Distance measure strategy: The process of calculating the distance as represented in [6] by calculating the q then deciding whether to use Euclidean distance or the Manhattan distance. Our proposed enhancement applied to improve the parallelization of the algorithm:

- Limiting the number of threads to be activated depending on whether the K quantity is small and requires reduced number of threads from the machine's maximum threads count to avoid the activation on unused threads.
- Instead of dividing the process of clusters and datapoints initialization into two separate sections, the improved parallel algorithm utilizes all the available threads in the initialization procedure to increase the speedup [18-19].

The source code of our improved parallel K-means algorithm can be found on GitHub [20]. A flowchart of the improved parallel K-means is demonstrated in Fig. 1.

IV. Results and Discussion

A series of experiments have been performed on the three K-means clustering algorithms to detect the improvements our approach has presented compared to the previous approaches. The results of the experiments have shown the substantial advantage found in our improved variation of the parallel K-means clustering approach in terms of cluster quality as well as the performance speed [21, 19]. The following subsections demonstrate each advantage found in our variation of the parallel K-means algorithm.

A. Cluster Quality

The tab. 1 and Tab. 2 shows that sequential and parallel have different numbers of points in each cluster in each run, but the improved parallel number of points in each cluster is stable in each run which enhances the cluster quality [22].

TABLE I. Sequential and Parallel Clustering Points

	Sequential	Parallel

Runs	C1	C2	C3	C1	C2	C3
1	4822	105587	89591	102346	4846	92808
2	4820	105953	89227	89318	105862	4820
3	4820	105924	89256	89227	105953	4820
4	4820	105863	89317	89221	105959	4820
5	89021	106159	4820	89763	4822	105415
6	4824	105266	89910	4845	91793	103362
7	4822	89591	105587	89289	105891	4820
8	4820	105961	89219	4827	104655	90518
9	4820	89220	105960	89220	4820	105960
10	4820	105856	89324	4824	89954	105222

TABLE II. IMPROVED PARALLEL CLUSTERING POINTS

	Improved Parallel		
Runs	**C1**	**C2**	**C3**
1	4820	89318	105862
2	4820	89318	105862
3	4820	89318	105862
4	4820	89318	105862
5	4820	89318	105862
6	4820	89318	105862
7	4820	89318	105862
8	4820	89318	105862
9	4820	89318	105862
10	4820	89318	105862

B. Improvement in Elapse Time

Tab. 3 shows the elapsed time of the sequential, parallel, Improved algorithm (without changing the parallelization on the parallel code), and the improved parallel (including the improved parallelization) on different number of clusters.

TABLE III. ELAPSED TIME FOR THREE ALGORITHMS

	K Clusters	Sequential	Parallel	Improved algorithm	Improved Parallel
Elapsed Time	**K = 5**	2.966	1.145	1.088	0.821
	K = 10	5.374	1.815	1.411	1.089
	K = 15	8.885	2.230	1.732	1.294
	K = 20	14.011	2.847	1.931	1.491
	K = 25	15.892	3.433	2.036	1.763
	K = 30	18.881	3.937	2.287	1.961

Table 4 shows the speedup of the improved algorithm as well as the improved parallel on different size of clusters [23, 24]. Moreover, it shows the percentage of improvement of elapsed time on the parallel code.

TABLE IV. IMPROVED PARALLEL SPEEDUP

	K Clusters	Improved algorithm speedup	Improved Parallel Speedup	Parallel vs Improved algorithm	Parallel vs Improved Parallel
Elapsed Time	**K = 5**	2.726	3.613	4.98%	24.54%
	K = 10	3.809	4.935	22.26%	22.82%
	K = 15	5.130	6.866	22.33%	25.29%
	K = 20	7.652	9.397	35.69%	18.57%
	K = 25	7.806	9.014	40.69%	13.41%
	K = 30	8.256	9.628	41.91%	14.25%

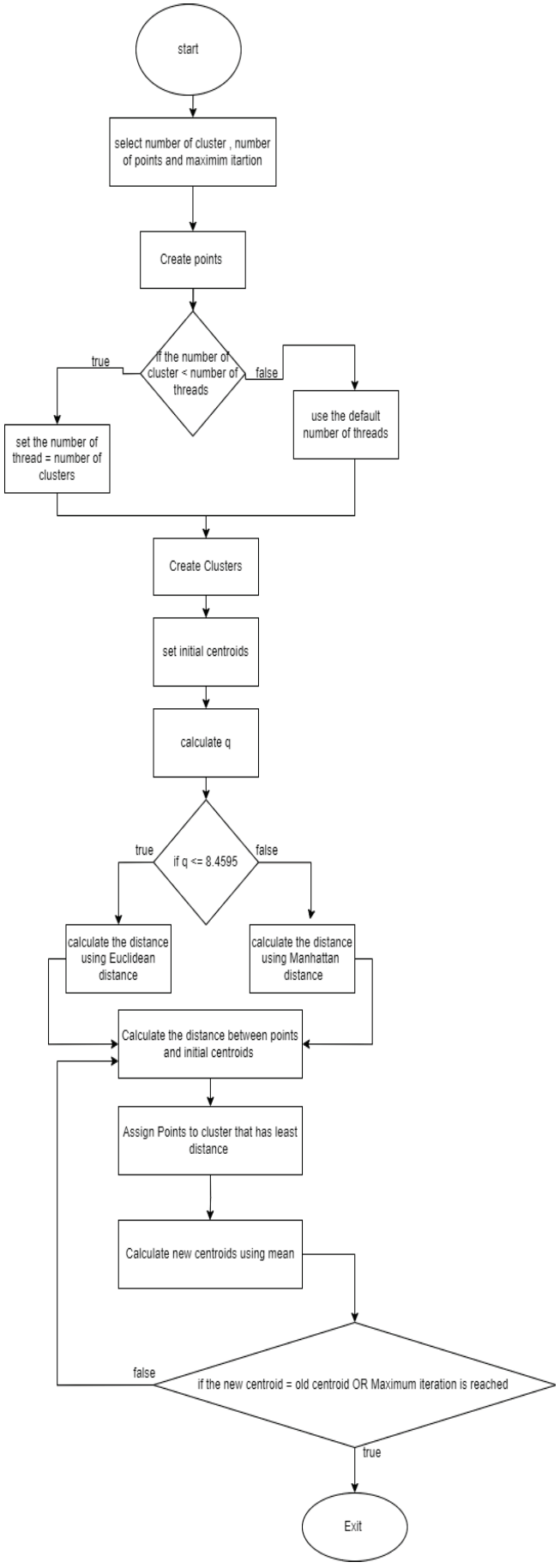

Fig. 1. Flowchart of improved parallel k-means clustering algorithm.

V. Conclusion & Future Work

The k-means clustering algorithm has numerous shortcomings. For example, the difficulty of dealing with voluminous data, the sensitivity of the outlier, and the random selection of the initial centroid. In this paper, we overcome these three limitations by cleaning the data prior to clustering to eliminate outliers, dividing the data into smaller sections using parallel programming, and selecting the initial centroid with care. In this paper, the enhanced parallel K-Means algorithm was applied to a dataset of 234 296 salary records. Due to the fact that the initial centroid was not calculated arbitrarily, the number of points in each cluster remained constant across multiple iterations, thereby improving cluster quality. In addition, the improved parallel K-means required less time to complete than the parallel and sequential k-means clustering algorithm. As part of future work, we aim to experiment with the performance of our improved k-means clustering algorithm as part of ensemble models. Future work could also extend to the application of the proposed algorithm in different domains, especially in the medical and cybersecurity areas.

References

[1] M. Gollapalli, “Literature review of attribute level and structure level data linkage techniques,” Int. J. Data Min. Knowl. Manag. Process, vol. 5, no. 5, pp. 01–20, 2015.

[2] M. Gollapalli, “Ensemble machine learning model to predict the waterborne syndrome,” Algorithms, vol. 15, no. 3, p. 93, 2022.

[3] M. Gollapalli and A. Maissa, “Task Failure Prediction Using Machine Learning Techniques in the Google Cluster Trace Cloud Computing Environment,” Mathematical Modelling of Engineering Problems, vol. 9, pp. 545–553, 2022.

[4] A. Alfaleh and M. Gollapalli, “A critical review of data mining techniques used for the management of sickle cell disease,” in Proceedings of the 12th International Conference on Computer Modeling and Simulation, 2020.

[5] Atta-Ur-Rahman et al., "Advance Genome Disorder Prediction Model Empowered With Deep Learning," in IEEE Access, vol. 10, pp. 70317-70328, 2022, doi: 10.1109/ACCESS.2022.3186998..

[6] Liao, Q., Yang, F., & Zhao, J. (2013, November). An improved parallel K-means clustering algorithm with MapReduce. In 2013 15th IEEE International Conference on Communication Technology (pp. 764-768). IEEE.

[7] Shang, R., Ara, B., Zada, I., Nazir, S., Ullah, Z., & Khan, S. U. (2021). Analysis of simple K-mean and parallel K-mean clustering for software products and organizational performance using education sector dataset. Scientific Programming, 2021, 1-20.

[8] Employee Payroll - Catalog. Available online https://catalog.data.gov/dataset/employee-payroll (accessed Feb. 14, 2023)

[9] M. Gollapalli, X. Li, and I. Wood, “Automated discovery of multi-faceted ontologies for accurate query answering and future semantic reasoning.” Data & Knowledge Engineering, vol. 87, pp. 405-424, 2013, doi: 10.1016/j.datak.2013.05.005.

[10] M. Gollapalli, L. Alabdullatif, F. Alsuwayeh, M. Aljouali, A. Alhunief and Z. Batook, "Text Mining on Hospital Stay Durations and Management of Sickle Cell Disease Patients," 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN), Al-Khobar, Saudi Arabia, 2022, pp. 1-6, doi: 10.1109/CICN56167.2022.10008265.

[11] M. Gollapalli et al., “A novel stacking ensemble for detecting three types of diabetes mellitus using a Saudi Arabian dataset: Pre-diabetes, T1DM, and T2DM,” Comput. Biol. Med., vol. 147, no. 105757, p. 105757, 2022.

[12] A. Sestini, “SestoAle/Parallel-K-Means: A parallel implementation of K-Means algorithm in C++ and OpenMP.” https://github.com/SestoAle/Parallel-K-Means (accessed Feb. 14, 2023)

[13] M. Gollapalli, X. Li, I. Wood, and G. Governatori, “Ontology Guided Data Linkage Framework for Discovering Meaningful Data Facts.” Advanced Data Mining and Applications, pp. 252-265, 2011, doi: 10.1007/978-3-642-25856-5_19

[14] M. Gollapalli, E. Al-Jaber, J. Selham, W. Al-awazem, Z. Al-Sayoud, and S. Al-Qassab, “Saudi Rural Breast Cancer Prevention Framework.” 2015 International Conference on Cloud Computing (ICCC), 2015, doi: 10.1109/cloudcomp.2015.7149653.

[15] M. Gollapalli et al., “Intelligent modelling techniques for predicting used cars prices in Saudi Arabia,” Mathematical Modelling of Engineering Problems, vol. 10, no. 1, 2023, pp. 139–148. doi: 10.18280/mmep.100115.

[16] N. Ahmed Sajid et al., “A Novel Metadata Based Multi-Label Document Classification Technique.” Computer Systems Science and Engineering, vol. 46, no. 2, pp. 2195-2214, 2023, doi: 10.32604/csse.2023.033844.

[17] M. Gollapalli et al., “Modeling Algorithms for Task Scheduling in Cloud Computing Using CloudSim.” Mathematical Modelling of Engineering Problems, vol. 9, no. 5, pp. 1201-1209, 2022, doi: 10.18280/mmep.090506.

[18] M. Gollapalli et al., “A Neuro-Fuzzy Approach to Road Traffic Congestion Prediction.” Computers, Materials & Continua, vol. 73, no. 1, pp. 295-310, 2022, doi: 10.32604/cmc.2022.027925.

[19] M. Jamal, N. Ahmad Zafar, Atta-ur-Rahman, D. Musleh, M. A. Gollapalli, and S. Chabani, “Modeling and Verification of Aircraft Takeoff Through Novel Quantum Nets.” Computers, Materials & Continua, vol. 72, no. 2, pp. 3331-3348, 2022, doi: 10.32604/cmc.2022.025205.

[20] L. Al-Ghamdi, Github, https://github.com/Lamaghamdi/Improved-Parallel-K-Means-Clustering (accessed Feb. 14, 2023).

[21] Atta-ur-Rahman et al., “Supervised Machine Learning-Based Prediction of COVID-19.” Computers, Materials & Continua, vol. 69, no. 1, pp. 21-34, 2021, doi: 10.32604/cmc.2021.013453.

[22] M. Gollapalli and A. Alfaleh, “An Artificial Intelligence Approach for Data Modelling Patients Inheritance of Sickle Cell Disease (SCD) in the Eastern Regions of Saudi Arabia.” Mathematical Modelling of Engineering Problems, vol. 9, no. 4, pp. 1079-1088, 2022, doi: 10.18280/mmep.090426.

[23] T.M. Ghazal, H. Al Hamadi, M. Umar Nasir, M. Gollapalli, M. Zubair, M. Adnan Khan, M., and C. Yeob Yeun, "Supervised machine learning empowered multifactorial genetic inheritance disorder prediction," Computational Intelligence and Neuroscience, 2022.

[24] M. Gollapalli, B. Saad, J. Alabdulkarim, R. Sendi, R. Alsabt and S. Alsharif, "Detection of Chronic Kidney Disease Using Machine Learning Approach," 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN), Al-Khobar, Saudi Arabia, 2022, pp. 460-465, doi: 10.1109/CICN56167.2022.10008293.