ISSN 2007-9737

Empirical Comparison of Scalability Strategies in Density-based Clustering Algorithms for Large Datasets

Adrián J. Ramírez-Díaz*, José Fco. Martínez-Trinidad, J. Ariel Carrasco-Ochoa

Instituto Nacional de Astrofísica,
Óptica y Electrónica, Computer Science,
Mexico

{aramirez, fmartine, ariel}@inaoep.mx

Abstract. Density-based clustering can discover clusters of arbitrary shapes while filtering noise, but its scalability for large datasets remains a challenge. Recent density-based clustering algorithms for large datasets employ different scalability strategies, such as parallelization, sampling, or reduction of the number of comparisons, to address this issue. This study compares density-based clustering algorithms for large datasets by explicitly evaluating the above-mentioned scalability strategies for processing large datasets under the same framework. Experiments were conducted on synthetic datasets scaled up to $50\times$. The results provide a comparative analysis that highlights that parallelism allows improving scalability, sampling reduces runtime with a potential loss in quality, and reducing comparisons between objects maintains clustering quality. These findings support the selection of algorithms according to available computational resources and dataset size, which is very valuable in practice.

Keywords. Clustering, large datasets, density-based clustering, scalability strategies

1 Introduction

Clustering is a fundamental unsupervised learning technique that groups objects into clusters according to their similarity [23, 11]. Among the different approaches, density-based clustering is widely used because it can discover clusters of arbitrary shapes while filtering noise [26, 20, 9]. However, as the volume of data increases, scaling density-based clustering algorithms remains a challenge [19].

Several scalability strategies for density-based clustering algorithms have been proposed [15]. Some of these algorithms utilize parallelization strategies by distributing the workload across multiple processing cores. Others employ sampling strategies to analyze only a representative subset of the dataset. Another strategy focuses on reducing the number of object-to-object comparisons to enhance the speed of the clustering process.

Previous works have compared density-based clustering algorithms for large datasets, focusing on runtime and clustering quality [21], listing the main characteristics of each algorithm [3], or evaluating their effectiveness in distributed architectures that exploit their parallelizable components [22, 14]. However, these studies do not explicitly evaluate how different scalability strategies, such as parallelism, sampling, or reducing the number of object-to-object comparisons, improve the scalability of density-based clustering algorithms under the same framework.

This work builds upon those studies and makes the following contributions:

— We evaluate the scalability of parallel algorithms (S-DBSCAN, K-DBSCAN, and RS-DBSCAN) by varying the number of processing cores.

— We analyze the effect of sample size in sampling-based algorithms (Improved KMeans and RS-DBSCAN) on clustering quality and runtime.

— We present a comparative framework that contrasts parallelism, sampling, and reduction of comparisons (KNN-BLOCK-DBSCAN) under the same experimental conditions.

The remainder of this paper is organized as follows. Section 2 reviews related work on scalability strategies for density-based clustering algorithms for large datasets. Section 3 reviews the algorithms under study. Section 4 describes the experimental design. Section 5.1 presents and discusses the results, and Section 6 concludes this work and outlines directions for future research.

2 Related Work

Density-based clustering, introduced with DBSCAN [8], has been widely studied due to its ability to handle noise and non-convex cluster shapes. However, DBSCAN is computationally expensive for large datasets, which has motivated the development of scalable algorithms. Existing works on density-based clustering identify three main scalability strategies to improve efficiency: reduction of object-to-object comparisons, parallelism, and sampling [4, 3, 16].

The process of identifying density-reachable objects to compute density is costly; therefore, reducing the number of comparisons between objects to identify neighborhoods improves the efficiency of density-based algorithms [4]. In parallel processing, data is distributed across multiple cores and processed simultaneously, which has led to algorithms that apply different partitioning, processing, and merging strategies [3, 14]. Sampling strategies reduce dataset size while attempting to preserve representative structures and subsequently propagate the clustering results to the rest of the dataset. Consequently, the quality of the result depends on the representativeness of the sample [16, 6].

Using these three strategies, several density-based clustering algorithms for large datasets have been proposed [4]. Comparative studies among these algorithms have also been reported [15]; however, most of these comparisons focus exclusively on parallel algorithms, since they can be implemented in distributed computing frameworks such as MapReduce and Apache Spark [22, 14], and therefore center the comparison on implementation and data-handling strategies in those architectures. Other works compare density-based clustering algorithms for large datasets [21], but do not include the effect of scalability strategies.

As far as we know, no study has conducted an empirical comparison of scalability strategies in density-based clustering algorithms for large datasets within a unified experimental framework. This motivates our study, which investigates the advantages and limitations of these scalability strategies.

3 Algorithms Under Study

This work analyzes five density-based clustering algorithms designed for large datasets. All of them are based on the density analysis and incorporate different strategies to improve scalability.

S-DBSCAN [17] is a parallel adaptation of DBSCAN. The algorithm randomly partitions the dataset, applies DBSCAN independently to each partition in parallel, and then merges the resulting clusters based on their centroid distances. Partitioning reduces the number of comparisons because each object is only compared with members of its partition, and all partitions can be processed simultaneously.

Improved KMeans [16] is a density-based algorithm that selects a random sample of the dataset, analyzes the sample to extract an initial set of clusters, and then assigns the remaining objects either to these clusters or to new clusters if needed. This reduces runtime but depends on the representativeness of the sample.

K-DBSCAN [10] also uses a parallel strategy, but the partitioning is not random. Instead, it partitions the dataset using the KMeans++ algorithm. Each partition is analyzed with DBSCAN in parallel. Then, cluster merging is determined by measuring the distance between their centroids and border points. The final combination of clusters is also performed with DBSCAN in parallel. This approach reduces the number of comparisons while preserving the DBSCAN result.

KNN-BLOCK-DBSCAN [7] reduces the complexity of density computation by minimizing the number of comparisons between objects. It builds a search structure that identifies neighboring points without comparing all objects in the dataset. The algorithm creates blocks of objects, classifies them as core, non-core, or noise, and then forms clusters by merging core blocks. This reduces redundant comparisons and reduces runtime.

RS-DBSCAN [6] integrates sampling and parallel processing. It extracts random samples, processes them with RNN-DBSCAN (a density-based clustering algorithm), and generates representative objects from the resulting clusters. These representative points are then clustered again using RNN-DBSCAN, and finally all objects in the dataset are assigned to the clusters derived from the representative objects. This combination leverages both parallelism and sample reduction.

The known time complexity of DBSCAN is $O(n^2)$ [18], and algorithms that build on it, such as S-DBSCAN and K-DBSCAN, reduce this cost to approximately $O((n/p)^2)$ by processing p partitions in parallel. RS-DBSCAN reports a complexity of $O(nr)$, where r represents the number of representative categories, whereas Improved KMeans has a complexity of $O(ns)$, with s denoting the size of the sample. KNN-BLOCK-DBSCAN reports quadratic complexity in the worst case; however, subquadratic behavior is expected in practice because its block-based KNN construction reduces the number of object-to-object distance evaluations. Algorithms that process independent partitions (S-DBSCAN, K-DBSCAN, and RS-DBSCAN) are suitable for distributed execution, while those based on sampling (enhanced KMeans, RS-DBSCAN) benefit from working on a smaller subset.

In summary, the density-based clustering algorithms for large datasets analyzed in this study are categorized according to the scalability strategies employed, as follows:

— Parallelism: S-DBSCAN, K-DBSCAN, RS-DBSCAN.

— Sampling: Improved KMeans, RS-DBSCAN.

— Reduction of comparisons: KNN-BLOCK-DBSCAN.

Table 1. Datasets and parameter settings used to evaluate density-based clustering algorithms

Dataset	Objects	Dim.	MinPts	ε
Flame	240	2	6	1.0
Pathbased	300	2	4	1.0
Jain	373	2	3	2.9
Compound	399	2	5	1.0
R15	600	2	3	0.3
Aggregation	788	2	6	1.21
2G_unbalance	1050	2	7	0.1
Complex8	2551	2	4	14.89
Complex9	3031	2	5	12.1
Cluto-t4-8k	8000	2	10	9.5

4 Experimental Design

This section describes the design of the experiments to evaluate the performance of density-based clustering algorithms for large datasets in terms of clustering quality and runtime.

4.1 Datasets

We conducted the experiments using a set of ten synthetic datasets obtained from the Tomas Barton repository [2], which have been widely used in previous works to compare density-based clustering algorithms. These datasets were selected because they include diverse clustering challenges representative of density-based analysis, such as clusters with different densities, non-convex shapes, and noise levels, as well as clear ground-truth labels that facilitate reproducible evaluation. Moreover, they have been commonly used in previous studies to evaluate density-based clustering algorithms. Table 1 summarizes the main characteristics of these datasets, including the name, number of objects, dimensionality, and the parameter values for ε and $minPts$ that, according to the literature, provide suitable clustering results with density-based algorithms [24, 10, 13].

To analyze the scalability of the algorithms, we increased the number of objects in each dataset by scaling them $10\times$, $20\times$, $30\times$, $40\times$, and $50\times$. Each additional object was generated by selecting an existing object as a reference and slightly perturbing its position at random. The magnitude of this perturbation was limited to a radius computed as

the average distance to the nearest neighbors of each selected object. In this way, the overall cluster structure of the dataset was preserved, while only the density of the regions was increased.

Although the original benchmarks contain up to $8{,}000$ objects, we utilized their scaled versions, which contain up to $400{,}000$ objects, a data size considered in the literature as a large dataset [6, 10, 16]. This dataset size enabled us to establish a computational bottleneck in our controlled environment. This was particularly important for evaluating the performance benefits and trade-offs associated with the scaling strategies used in algorithms that exhibit quadratic complexity ($O(n^2)$).

4.2 Evaluation Metrics

Two external clustering quality metrics were employed, both requiring ground-truth labels: Normalized Mutual Information (NMI) [25] and Adjusted Rand Index (ARI) [12]. These metrics are commonly used in the literature to evaluate density-based clustering algorithms [5, 1]. For runtime experiments, execution times were measured in seconds, excluding input/output operations such as data loading and result storage.

4.3 Parameter Setting

Each algorithm under study was evaluated using the parameters described below:

— S-DBSCAN and K-DBSCAN: Both algorithms use three parameters: ε, $minPts$, and the number of partitions. For ε and $minPts$, we adopted the values reported in the literature (see Table 1). The number of partitions was varied across experiments with values $\{2, 5, 10, 15, 20\}$.

— Improved KMeans (IKMEANS) and KNN-BLOCK-DBSCAN: These algorithms also rely on ε and $minPts$, for which we used the values specified in Table 1. In addition, IKMEANS requires the sample size as a parameter. We evaluated the algorithm using sample sizes corresponding to 10%, 20%, 30%, 40%, and 50% of the dataset.

— RS-DBSCAN: Unlike the previous algorithms, RS-DBSCAN does not use ε or $minPts$. Instead, it requires two parameters, k_1 and k_2, to determine density. Following the authors' recommendation [6], both parameters were set equal to $minPts$. RS-DBSCAN also requires the sample size and the number of partitions. For these, we adopted the same criteria as those applied to IKMEANS (sample size) and to S-DBSCAN/K-DBSCAN (number of partitions).

5 Results and Discussion

This section presents the results of the experimental evaluation. We analyze the effect of parallelism and sampling on clustering quality and runtime. For clarity, results are reported as averages across all datasets for each configuration of parameters. Quality was measured using the NMI and ARI metrics, and runtime is reported in seconds. For a fair comparison between the analyzed algorithms, we implemented all of them in Python. The experiments were performed on a computer with two Intel Xeon E5-2620 processors at 2.40 GHz, 256 GB of RAM, and Linux Ubuntu 22.

5.1 Effect of Parallelism

Three algorithms incorporate parallelism: K-DBSCAN, RS-DBSCAN, and S-DBSCAN. Figures 1 and 2 show their average quality and runtime, respectively. In the plots, the x-axis corresponds to the dataset scaling factor ($10\times$, $20\times$, $30\times$, $40\times$, and $50\times$), and within each scaling factor, different numbers of cores are shown (2, 5, 10, 15, and 20). Each line corresponds to one algorithm: K-DBSCAN (red), RS-DBSCAN (blue), and S-DBSCAN (green).

Regarding clustering quality (Fig. 1), all three algorithms show consistent results across parameter combinations, with values around 0.6 on average. S-DBSCAN outperforms K-DBSCAN, which in turn surpasses RS-DBSCAN. However, neither the number of cores nor the increase in dataset size produced noticeable changes in clustering quality.

In contrast, runtime (Fig. 2) is highly affected by dataset size; the runtime growth is approximately

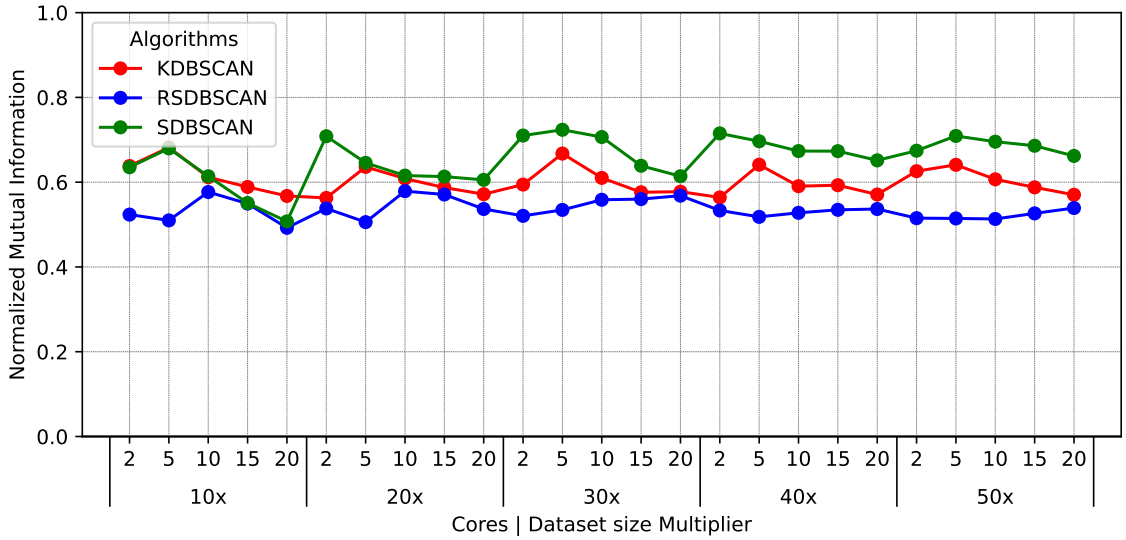

Fig. 1. Clustering quality mean (NMI) of parallel algorithms across dataset scales and number of cores

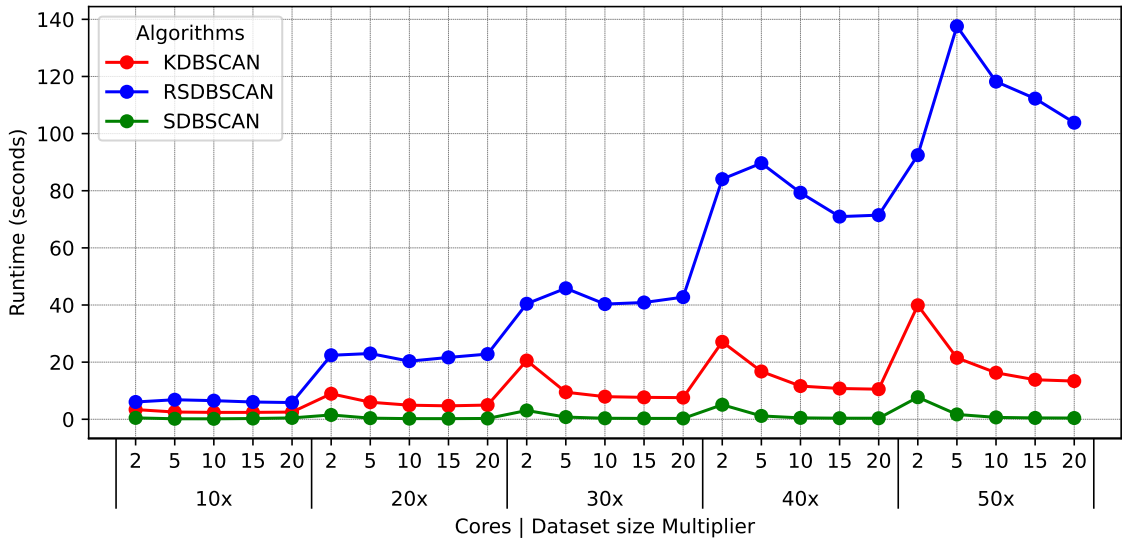

Fig. 2. Runtime mean of parallel algorithms across dataset scales and number of cores

quadratic, as expected in density-based algorithms. From 2 to 5 cores, there is the largest runtime reduction, followed by a smaller but still notable one from 5 to 10 cores.

The gain decreases from 10 to 15 cores, and the smallest decreases from 15 to 20 cores. These results indicate that increasing the number of cores beyond a certain point does not substantially reduce runtime. Among the algorithms, RS-DBSCAN required the highest runtime, followed by K-DBSCAN, and S-DBSCAN achieved the lowest runtime. Despite these differences, the three algorithms consistently exhibited the same expected parallel behavior.

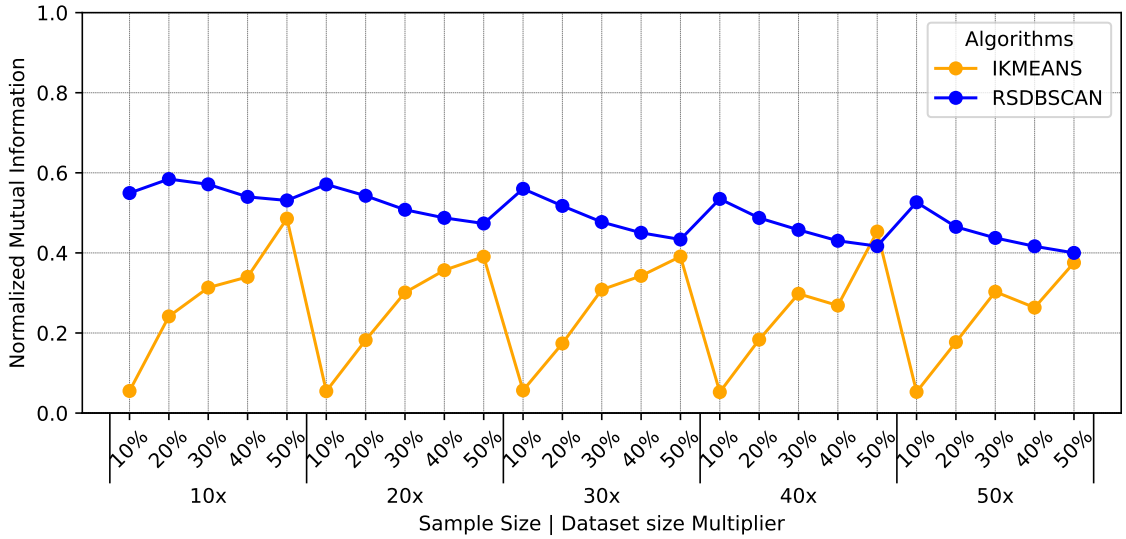

Fig. 3. Clustering quality mean (NMI) of sampling-based algorithms across dataset scales and number of cores

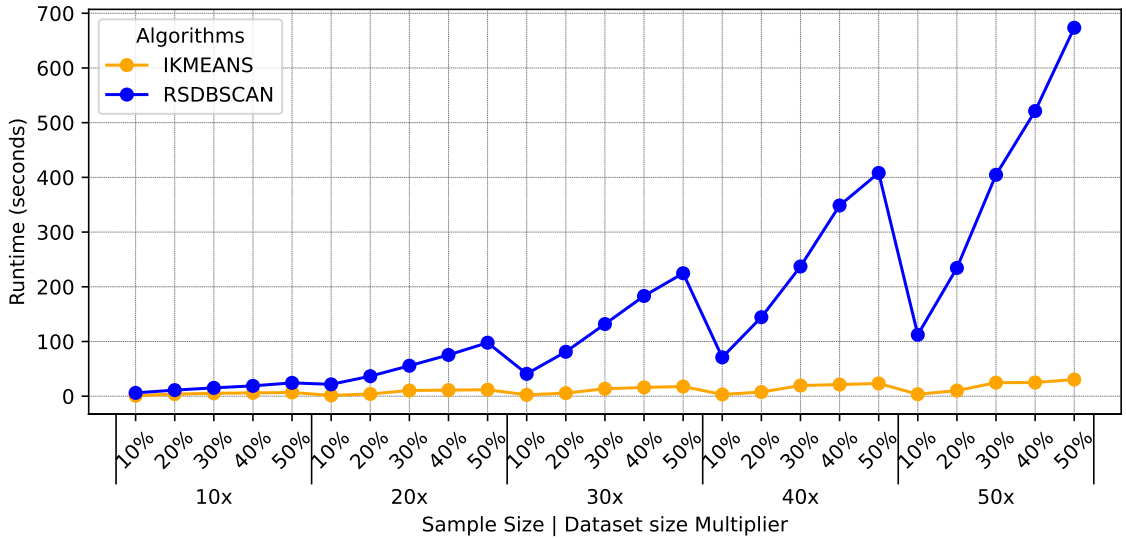

Fig. 4. Runtime mean of sampling-based algorithms across dataset scales and number of cores

5.2 Effect of Sampling

Figures 3 and 4 show the results of the sampling-based algorithms, IKMEANS (orange) and RS-DBSCAN (blue). In these experiments, the x-axis corresponds to the dataset scaling factor, and within each factor, different sample sizes are shown as percentages of the dataset (10%, 20%, 30%, 40%, and 50%). In contrast to the parallel algorithms, which vary the number of partitions or processing cores, this scalability strategy varies the fraction of the dataset used as the sample.

For clustering quality (Fig. 3), IKMEANS shows stable behavior across dataset sizes, indicating

that the number of objects does not affect quality. However, the quality strongly depends on sample size: values are close to zero for 10% samples and increase to around 0.6 for 50% samples. RS-DBSCAN shows a slight decrease in quality as dataset size increases, but this effect is minor. The main trend is also linked to sample size: larger samples yield lower quality, which indicates that the process of extracting information from the sample is affected by redundant data. In almost all cases, RS-DBSCAN consistently achieves higher quality compared to IKMEANS.

Regarding runtime (Fig. 4), both algorithms exhibit increasing runtimes as sample size grows, consistent with the higher computational effort of processing larger samples. The increase in runtime follows a quadratic trend, consistent with the density-based clustering algorithms applied to each sample. RS-DBSCAN required more runtime than IKMEANS; however, both algorithms demonstrated the same tendency: runtime increases as sample size grows.

5.3 Evaluation of Scalability Strategies in Density-Based Clustering for Large Datasets

KNN-BLOCK-DBSCAN relies on reducing the number of object-to-object comparisons and does not use sampling or parallelism. To compare it fairly with the other algorithms, we considered each algorithm under its best-performing configuration: IKMEANS with 50% sampling, S-DBSCAN and K-DBSCAN with 20 cores, and RS-DBSCAN with 20 cores and 10% sampling. Figures 5 and 6 show the results in terms of ARI and runtime, respectively, across dataset sizes.

Regarding runtime (Fig. 6), KNN-BLOCK-DBSCAN was the slowest algorithm, reaching over 8000 seconds, being unable to process datasets beyond the $40\times$ scale. This result indicates that reducing comparisons alone does not guarantee lower runtime for large datasets. In contrast, S-DBSCAN achieved the lowest runtime among all algorithms.

In terms of clustering quality (Fig. 5), KNN-BLOCK-DBSCAN consistently achieved the highest ARI values, outperforming all other algorithms. This suggests that the scalability strategy of reducing object-to-object comparisons better preserves cluster structure compared to sampling or partitioning strategies, but with a longer runtime. S-DBSCAN reached the second-best quality, showing that parallel scalability strategies reduced runtime while only moderately affecting clustering quality.

Improved KMeans, although it has often been reported as the fastest algorithm in prior studies, required larger samples (50%) to achieve quality comparable to that of K-DBSCAN and RS-DBSCAN. Under these conditions, its runtime was similar to that of the other algorithms, indicating that the sample size plays an important role in balancing quality and runtime.

Statistical analysis using one-way ANOVA and post-hoc tests ($\alpha = 0.05$) showed that KNN-BLOCK-DBSCAN exhibited significantly longer runtimes than the other algorithms ($p < 0.001$). For clustering quality, KNN-BLOCK-DBSCAN and S-DBSCAN obtained the highest scores, with statistically significant differences between KNN-BLOCK-DBSCAN and Improved KMeans, RS-DBSCAN, and K-DBSCAN ($p < 0.001$). No significant difference in quality was observed between KNN-BLOCK-DBSCAN and S-DBSCAN ($p = 0.088$). These results indicate that S-DBSCAN achieved clustering quality comparable to KNN-BLOCK-DBSCAN while requiring substantially lower runtime.

To summarize the findings, Table 2 links each scalability strategy to the corresponding algorithms and the observed trade-offs between clustering quality and runtime.

6 Conclusion and Future Work

This work presented an empirical evaluation of five density-based clustering algorithms for large datasets, each incorporating different scalability strategies: parallelism (S-DBSCAN, K-DBSCAN, RS-DBSCAN), sampling (Improved KMeans, RS-DBSCAN), and object-to-object comparison reduction (KNN-BLOCK-DBSCAN). The experiments were conducted on synthetic datasets scaled up to $50\times$, varying parameters such as the number of cores and sample size.

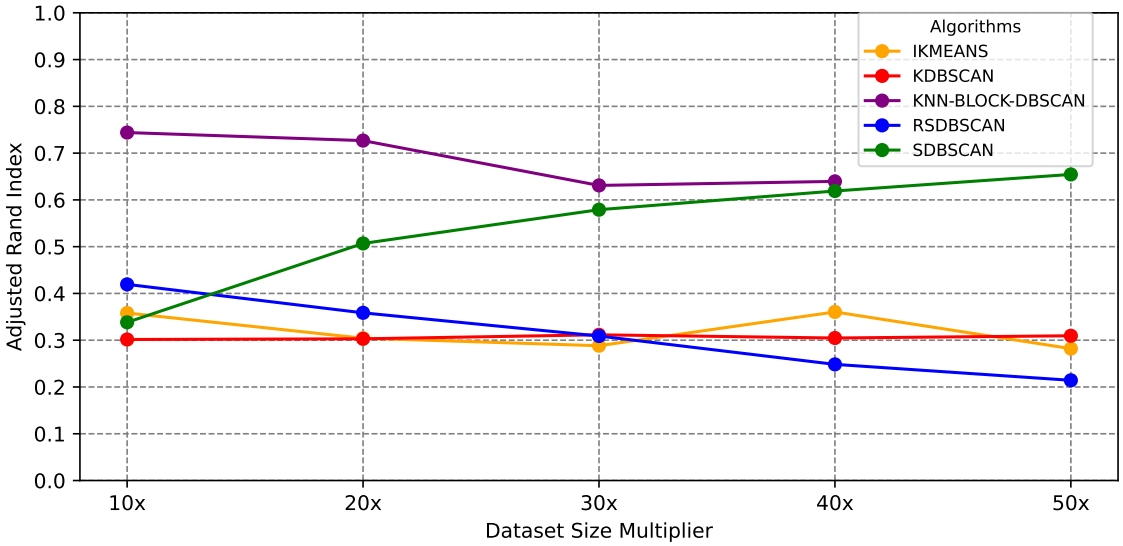

Fig. 5. Clustering quality mean (ARI) of density-based clustering algorithms across dataset scales

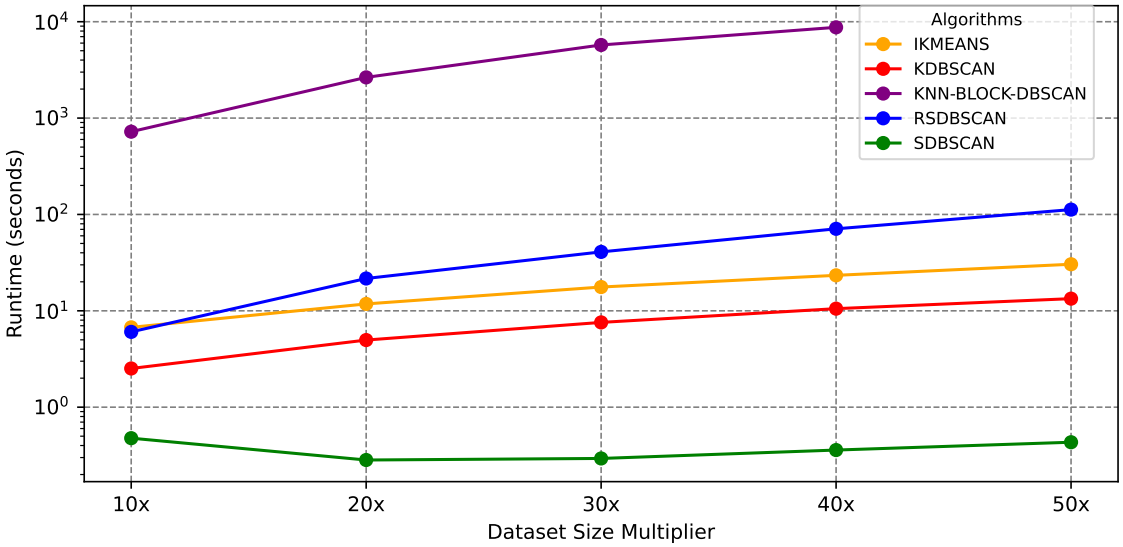

Fig. 6. Clustering runtime mean of density-based clustering algorithms across dataset scales

From our experiments, we can conclude that parallel scalability strategies effectively reduce runtime without introducing statistically significant losses in clustering quality. Specifically, S-DBSCAN achieved the lowest runtime while maintaining clustering quality comparable to KNN-BLOCK-DBSCAN, indicating that parallel partitioning successfully balances quality and runtime. Conversely, sampling scalability strategies achieve faster runtime but are highly sensitive to sample size: small samples produce poor clustering quality, while larger samples improve quality at the

Table 2. Summary of scalability strategies, algorithms, and observed trade-offs between clustering quality and runtime

Scalability strategy	Algorithms	Quality	Runtime
Parallelism	S-DBSCAN, K-DBSCAN, RS-DBSCAN	High–moderate	Lowest
Sampling	Improved KMeans, RS-DBSCAN	Moderate–variable	Low (depends on sample size)
Comparison reduction	KNN-BLOCK-DBSCAN	Highest	Highest

cost of quadratic runtime growth. RS-DBSCAN consistently achieved higher clustering quality compared to improved K-means, but it also requires longer runtime. Finally, KNN-BLOCK-DBSCAN, based on a reduced object-to-object comparison scalability strategy, obtained the highest clustering quality but was the slowest algorithm.

Overall, our study indicates that each scalability strategy involves trade-offs between quality and runtime. Parallelism is the most effective for large-scale scenarios when multiple cores are available, sampling can be useful when approximate solutions are acceptable, and comparison reduction preserves quality at the expense of runtime. These findings offer insights for selecting algorithms depending on dataset size and available computational resources, which is very valuable in practice.

Future work should focus on developing new density-based clustering algorithms that integrate parallelism, sampling, and comparison-reduction strategies. This approach could be explored to achieve better scalability while preserving clustering quality.

Acknowledgments

This work was financially supported by the Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) through the scholarship grant 778974.

References

1. **Amroune, N., Benazi, M., Sayad, L. (2024).** An adaptative eps parameter of dbscan algorithm for identifying clusters with heterogeneous density. Computación y Sistemas, Vol. 28, No. 2, pp. 465–472. DOI: 10.13053/cys-28-2-4600.

2. **Barton, T. (2019).** Clustering-benchmark. https://github.com/deric/clustering-benchmark.

3. **Bhattacharjee, P., Mitra, P. (2020).** A survey of density based clustering algorithms. Frontiers of Computer Science, Vol. 15, No. 1, pp. 151308. DOI: 10.1007/s11704-019-9059-3.

4. **Bushra, A. A., Yi, G. (2021).** Comparative analysis review of pioneering dbscan and successive density-based clustering algorithms. IEEE Access, Vol. 9, pp. 87918–87935. DOI: 10.1109/ACCESS.2021.3089036.

5. **Chen, X., Wang, P. (2025).** Three-way clustering based on digital image processing. IEEE Access, Vol. 13, pp. 105211–105219. DOI: 10.1109/ACCESS.2025.3579784.

6. **Chen, Y., Yang, Y., Pei, S., Chen, Y., Du, J. (2024).** A simple rapid sample-based clustering for large-scale data. Engineering Applications of Artificial Intelligence, Vol. 133, pp. 108551. DOI: https://doi.org/10.1016/j.engappai.2024.108551.

7. **Chen, Y., Zhou, L., Pei, S., Yu, Z., Chen, Y., Liu, X., Du, J., Xiong, N. N. (2021).** Knn-block dbscan: Fast clustering for large-scale data. IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 51, pp. 3939–3953.

8. **Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996).** A density-based algorithm for discovering clusters in large spatial databases with noise. kdd, Vol. 96, No. 34, pp. 226–231.

9. **Gallardo García, R., Beltrán, B., Vilariño, D., Zepeda, C., Martínez, R. (2020).** Comparison of clustering algorithms in text clustering tasks. Computación y Sistemas, Vol. 24, No. 2, pp. 429–437.

10. **Gholizadeh, N., Saadatfar, H., Hanafi, N. (2021).** K-dbscan: An improved dbscan algorithm for big data. The Journal of Supercomputing, Vol. 77, No. 6, pp. 6214–6235.

11. **Giordani, P., Ferraro, M. B., Martella, F. (2020).** Introduction to clustering. In An Introduction to Clustering with R. Springer, pp. 3–5.

12. **Halkidi, M., Batistakis, Y., Vazirgiannis, M. (2002).** Cluster validity methods: part i. ACM Sigmod Record, Vol. 31, No. 2, pp. 40–45.

13. **Hanafi, N., Saadatfar, H. (2022).** A fast dbscan algorithm for big data based on efficient density calculation. Expert Systems with Applications, Vol. 203, pp. 117501.

14. **Khader, M., Al-Naymat, G. (2020).** Density-based algorithms for big data clustering using mapreduce framework: A comprehensive study. ACM Comput. Surv., Vol. 53, No. 5, pp. 1–38. DOI: 10.1145/3403951.

15. **Kulkarni, O., Burhanpurwala, A. (2024).** A survey of advancements in dbscan clustering algorithms for big data. 2024 3rd International conference on Power Electronics and IoT Applications in Renewable Energy and its Control (PARC), pp. 106–111. DOI: 10.1109/PARC59193.2024.10486339.

16. **Lu, W. (2020).** Improved k-means clustering algorithm for big data mining under hadoop parallel framework. Journal of Grid Computing, Vol. 18, No. 2, pp. 239–250.

17. **Luo, G., Luo, X., Gooch, T. F., Tian, L., Qin, K. (2016).** A parallel dbscan algorithm based on spark. 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), pp. 548–553. DOI: 10.1109/BDCloud-SocialCom-SustainCom.2016.85.

18. **Luo, R., Li, T., Pu, R., Yang, J., Tang, D., Yang, L. (2025).** Nagb-dbscan: An improved dbscan clustering algorithm by natural neighbor and granular-ball. Information Sciences, Vol. 719, pp. 122445. DOI: https://doi.org/10.1016/j.ins.2025.122445.

19. **Muthu, S., Kalyan, G. R., Samuthira Pandi, V., D, S., J, L. P., S, J. (2025).** Big data clustering algorithms: Improving efficiency and scalability for massive data mining and pattern recognition. 2025 International Conference on Pervasive Computational Technologies (ICPCT), pp. 991–996. DOI: 10.1109/ICPCT64145.2025.10940783.

20. **Nalawade, S., Gokhale, S., Ingale, S., Arora, S., Jahirabadkar, S. (2023).** Hybrid density- grid based clustering algorithms: A review. 2023 7th International Conference On Computing, Communication, Control And Automation (ICCUBEA), pp. 1–5. DOI: 10.1109/ICCUBEA58933.2023.10392256.

21. **Ramírez-Díaz, A. J., Martínez-Trinidad, J. F., Carrasco-Ochoa, J. A. (2025).** Empirical comparison of density-based clustering algorithms for large datasets. **López-Monroy, A. P., Rosales-Pérez, A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., Olvera-López, J. A.**, editors, Pattern Recognition, Springer Nature Switzerland, Cham, pp. 46–55.

22. **Sardar, T. H. (2024).** Reflecting on a decade of evolution: Mapreduce-based advances in partitioning-based, hierarchical-based, and density-based clustering (2013–2023). WIREs Data Mining and Knowledge Discovery, Vol. 14, No. 6, pp. e1566. DOI: https://doi.org/10.1002/widm.1566.

23. **Wang, Y., Qian, J., Hassan, M., Zhang, X., Zhang, T., Yang, C., Zhou, X., Jia, F. (2024).** Density peak clustering algorithms: A review on the decade 2014–2023. Expert Systems with Applications, Vol. 238, pp. 121860. DOI: https://doi.org/10.1016/j.eswa.2023.121860.

24. **Wang, Y., Wang, D., Zhou, Y., Zhang, X., Quek, C. (2023).** Vdpc: Variational density peak clustering algorithm. Information Sciences, Vol. 621, pp. 627–651. DOI: https://doi.org/10.1016/j.ins.2022.11.091.

25. Zhang, P. (2015). Evaluating accuracy of community detection using the relative normalized mutual information. Journal of Statistical Mechanics: Theory and Experiment, Vol. 2015, No. 11, pp. P11006.

26. Zou, Y., Wang, Z., Wang, X., Lv, T. (2025). A clustering algorithm based on local relative density. Electronics, Vol. 14, No. 3, pp. 481. DOI: 10.3390/electronics14030481.

Article received on 08/09/2025; accepted on 28/11/2025.
**Corresponding author is Adrián J. Ramírez-Díaz.*