Classification using Spark- enabled Swarm Intelligence Algorithms

North Dakota State University - Universidad de Chile

By: Angeles Marin Batana

Abstract

Machine learning is a rapidly advancing discipline in computational science due to the increasing demand for the analysis and interpretation of large amounts of data from various sources and consequently, many optimization algorithms have been proposed to enhance performance and accuracy in machine learning models, particularly classification models.

This study investigates the performance improvements of 2 Bat Algorithm implementations when parallelized using Apache Spark, measuring efficiency gains from distributing data across partitions and distributing particles across partitions. The speedup and scaleup characteristics of these implementations were evaluated across different core configurations and data sizes, with results indicating diminishing returns with increasing processors and data sizes for both approaches, highlighting improvements for optimized strategies to achieve enhanced scalability.

Introduction

Optimization techniques are crucial in machine learning for minimizing error functions, tuning model parameters, and finding optimal solutions. Optimization algorithms like Particle Swarm Optimization (PSO), developed by Russell Eberhart and James Kennedy in 1995, are widely used nature-inspired algorithms that simulate the behavior of social animals. These algorithms use exploration and exploitation techniques to find the most optimal solutions in complex optimization problems in machine learning tasks.

The Bat Algorithm (BA), introduced by Xin-She Yang in 2010, builds on these principles by incorporating the echolocation behaviors of bats, as shown in Figure 1. In the BA, each bat represents a potential solution to the optimization problem, and a population of n bats is used to explore the search space. Initially, bats emit louder pulses at lower rates to gather information from wider areas. As they get closer to their target, they reduce their loudness and increase the pulse rate, allowing for a more focused search and convergence towards the global best solution. If a new solution is better than the current best, it replaces the current best and becomes the new global best. This process continues until the maximum number of iterations is reached.

This study focuses on the parallelization of the BA using Apache Spark to handle large datasets efficiently and improve computational performance.

```
    Objective function Obj(X) , $X=[x_1,x_2,\dots,x_{nv}]^T$
    Begin
    Initialize the bat population $x_i$ and $v_i$ (i=1,2,..,n)
    Define pulse frequency of $f_i$ at $x_i$
    Initialize pulse rates $r_i$ and the loudness $A_i$
While (t<maximum number of iterations)
        Generate new solutions by adjusting frequency and update velocities and
     positions (equation 1, 2 and 3)
        If (rand >$r_i$)
            Select a solution among the best solutions randomly;
            Generate a local solution around the selected best solution by a local
        random walk (equation 4)
        End if
        If (rand<$A_i$ and $f(x_i)<f(x^{cgbest})$)
            Accept the new solution
            Increases $r_i$ and decrease $a_i$
        End if
            Rank the bats at each iteration and store their current global best $x^{cgbest}$
     End while
    Post processing the results
    End
```

Figure 1: BA Pseudo Code

Methodology

A dataset containing features and target variables for classification is replicated to create 10 different dataset with sizes ranging from 200 to 2000 repetitions, in increments of 200. These datasets are tested across 7 different core configurations (2, 4, 8, 16, 24, 32, 64 cores) to analyze the parallel processing performance of 2 parallelization strategies:

1. Distributing Data Across Partitions: In this approach, the dataset is read into an RDD and the RDD is partitioned across multiple cores. Each partition of the RDD contains a subset of the data and the BA is applied independently to each data partition, leveraging data parallelism where the algorithm runs concurrently on different subsets of the data.

2. Distributing Particles (Bats) Across Partitions: In this approach, the population of particles, or bats, is partitioned into sub-populations, and this RDD of sub-populations is distributed across multiple cores. This strategy focuses on task parallelism, where different subsets of bat populations run and perform the Bat Algorithm concurrently.

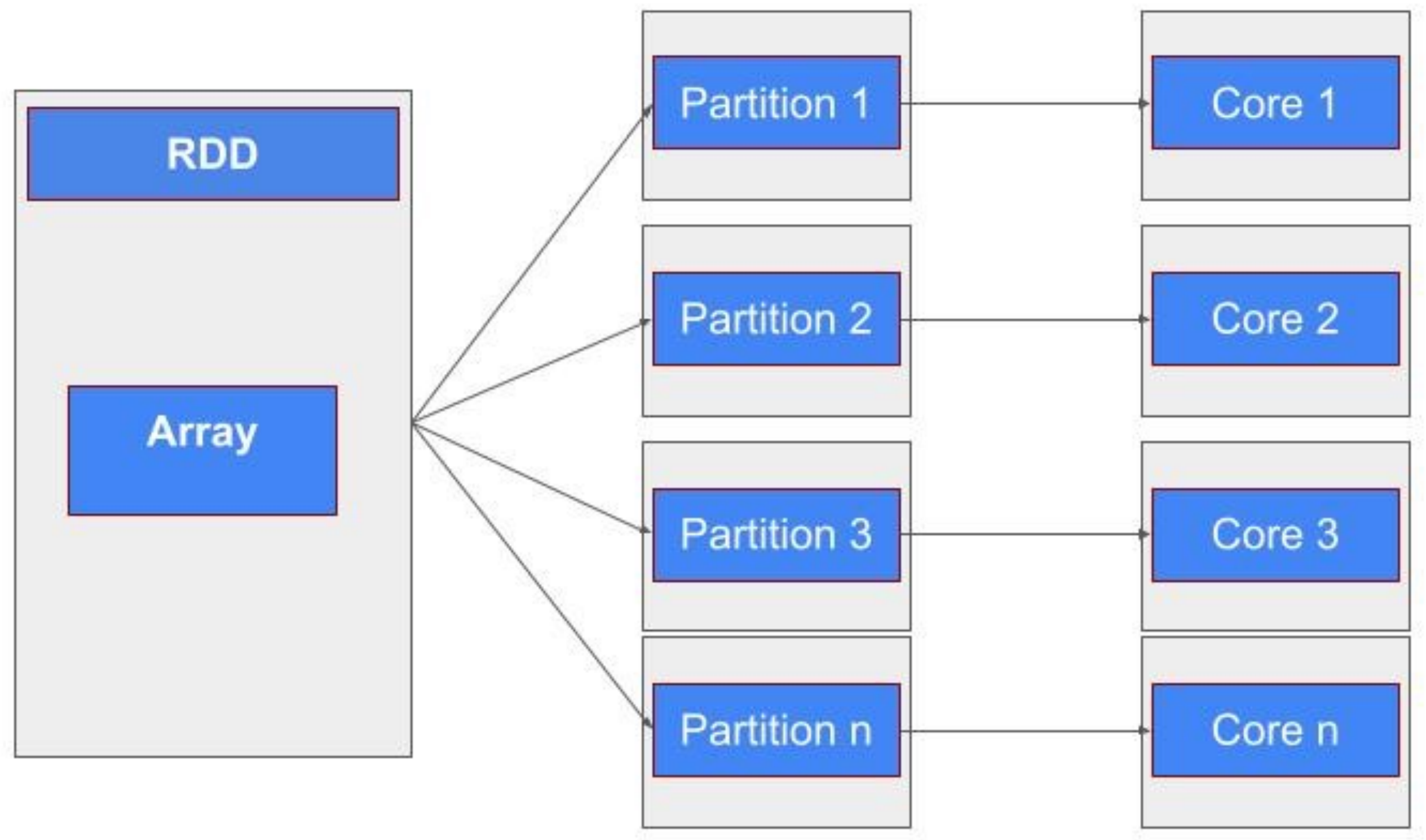

Figure 2: Parallelizing Operations Across Multiple Cores

Results

The performance of both parallelization methods across different datasets and core configurations were evaluated using speedup and scaleup metrics.

Speedup Analysis

Speedup plots were used to measure how the execution time of the programs decreased as the number of processors increased, achieving ideal parallel efficiency with faster processing times when more processors are added. Speedup was calculated by dividing the execution time with 2 processors by the execution time with *N* processors. Ideally, the speedup plot would approach a linear trend, where doubling the number of processors approximately halves the execution time.

Distributed Data Plot: Figure 3 demonstrates initial gains for the speedup curves, showing significant improvements when increasing from 2 to 16 cores. The speedup gains then plateau, with minimal improvements beyond 24 cores.

Distributed Particles Plot: Figure 4 shows the speedup curves following a similar trend to the data distribution approach, demonstrating substantial initial improvements up to about 24 cores, then plateauing with diminishing returns.

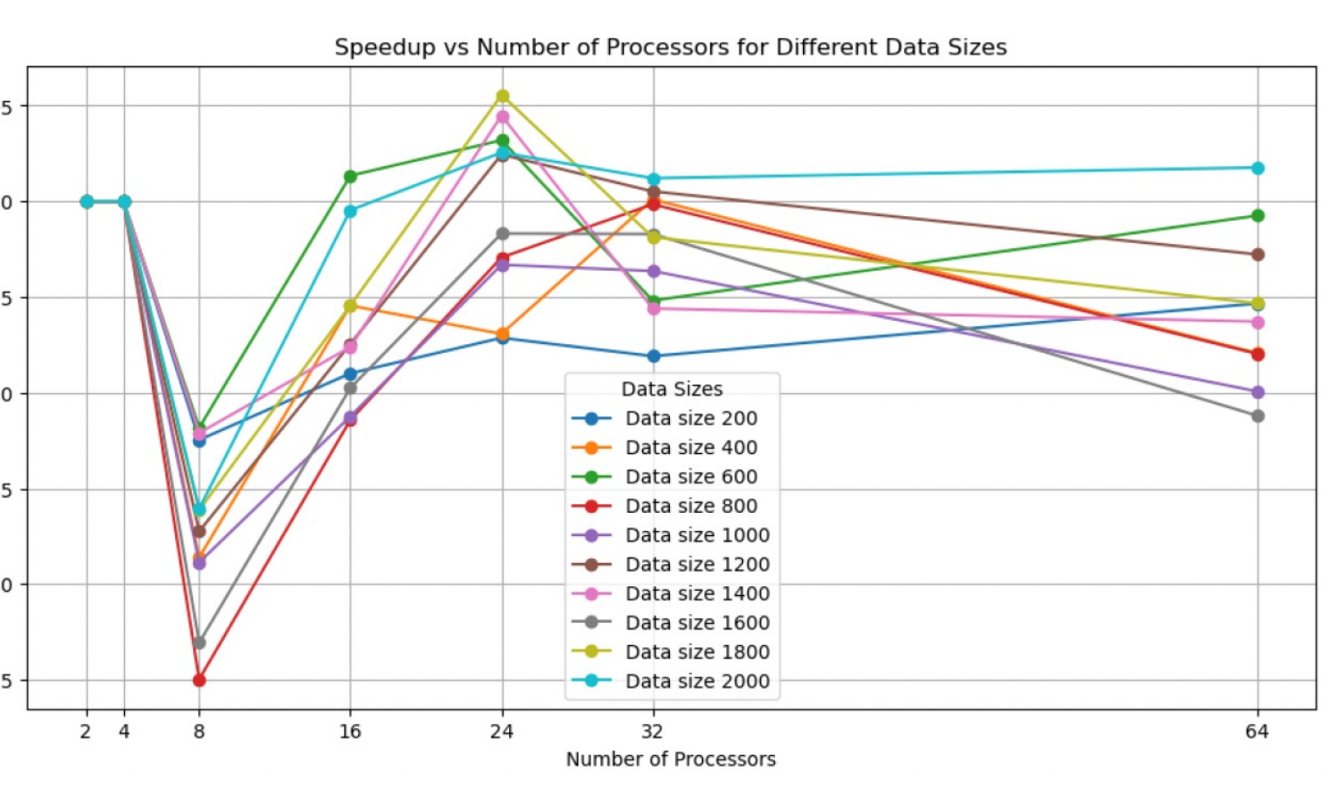

Figure 3: Speedup Plot for Distributed Data

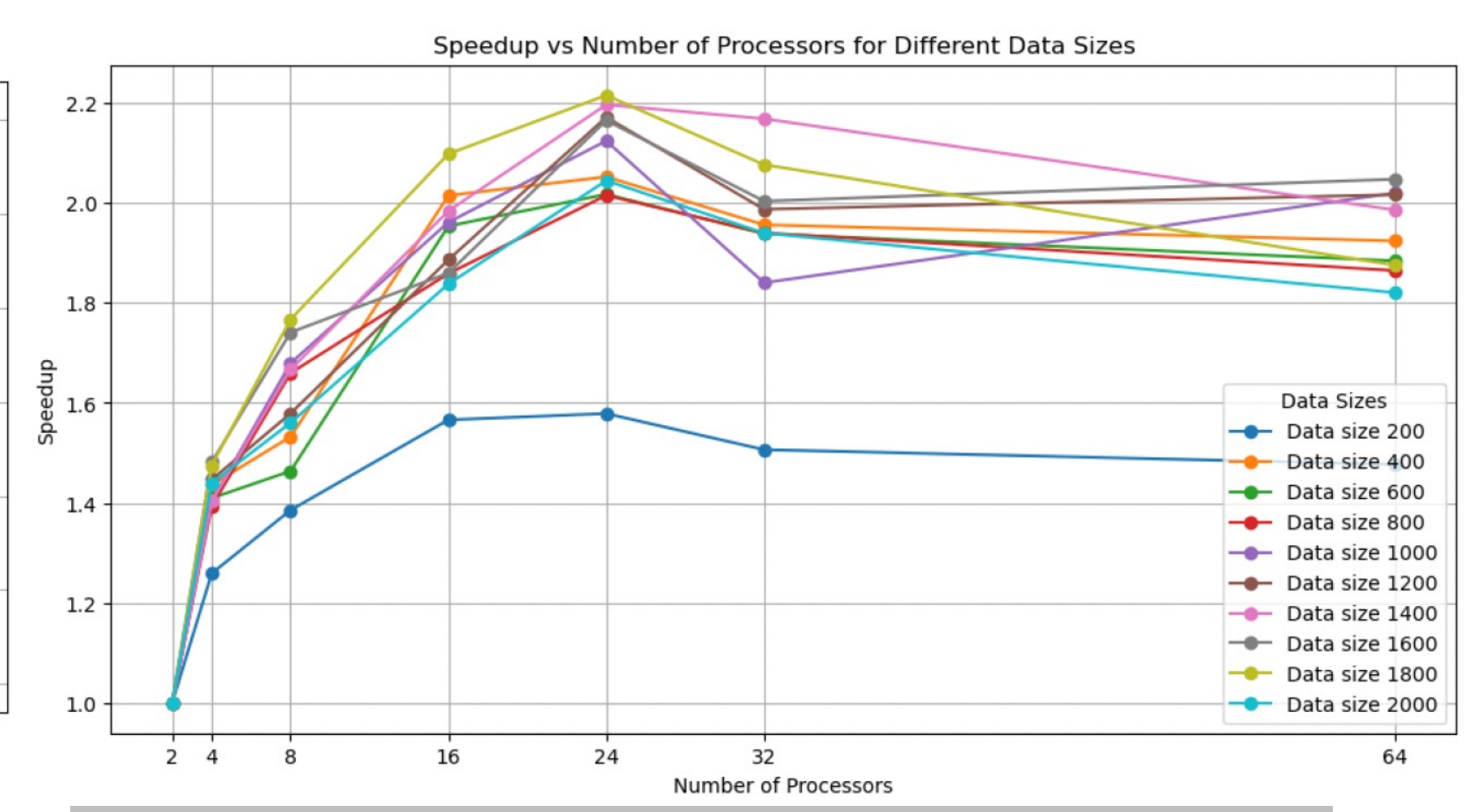

Figure 4: Speedup Plot for Distributed Particles

Scaleup Analysis

The scaleup plot measures how the execution time changes as the size of the dataset increases for a given number of processors used. It was calculated by dividing the execution time for a given data size by the smallest data size where, ideally, the execution time remains consistent as the data size increases, showing that the data size does not significantly impact processing time.

Distributed Data Plot: Figure 5 demonstrates promising scalability for datasets up to about 1000 repetitions, after which the plot begins to increase significantly, indicating longer processing times for larger datasets.

Distributed Particles Plot: Comparable to the data distribution approach, figure 6 shows higher core counts improved scale-up but were still affected by scalability limitations for larger datasets.

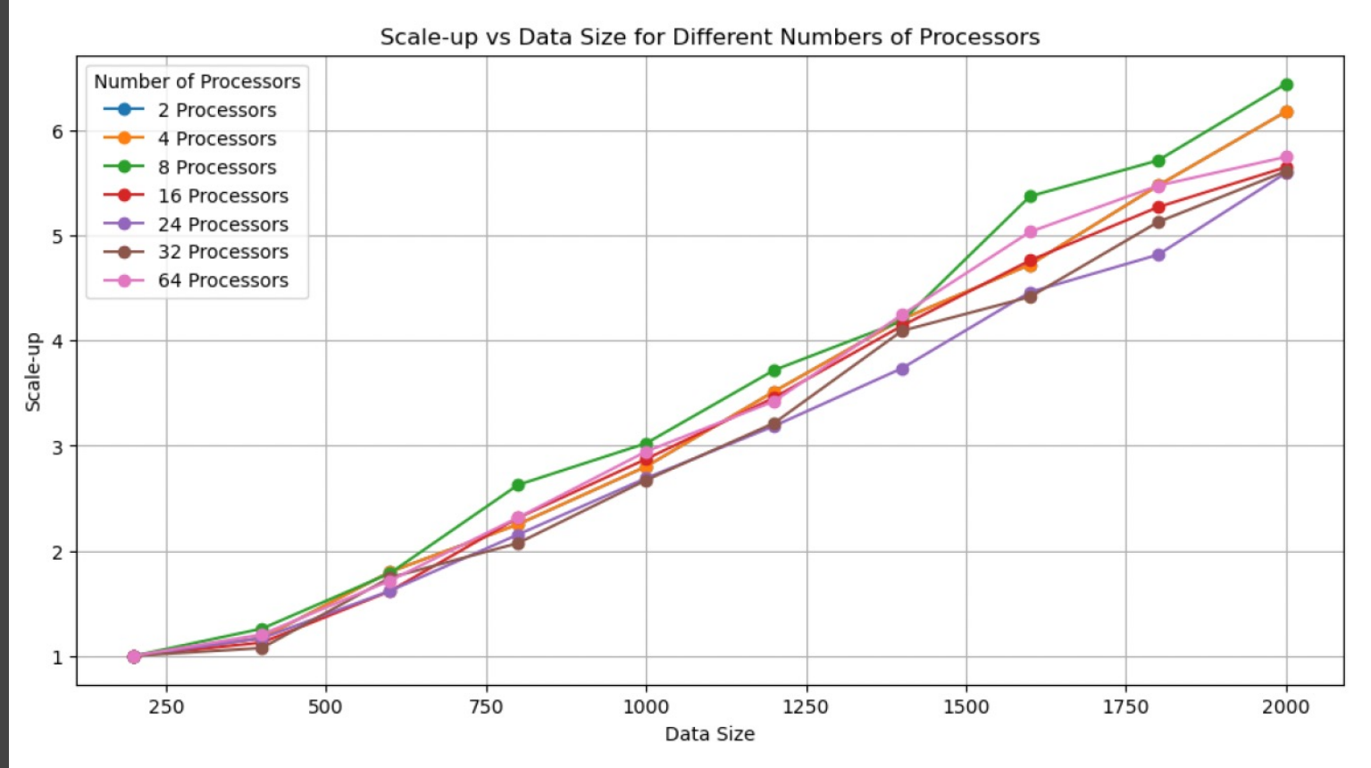

Figure 5: Scaleup Plot for Distributed Data

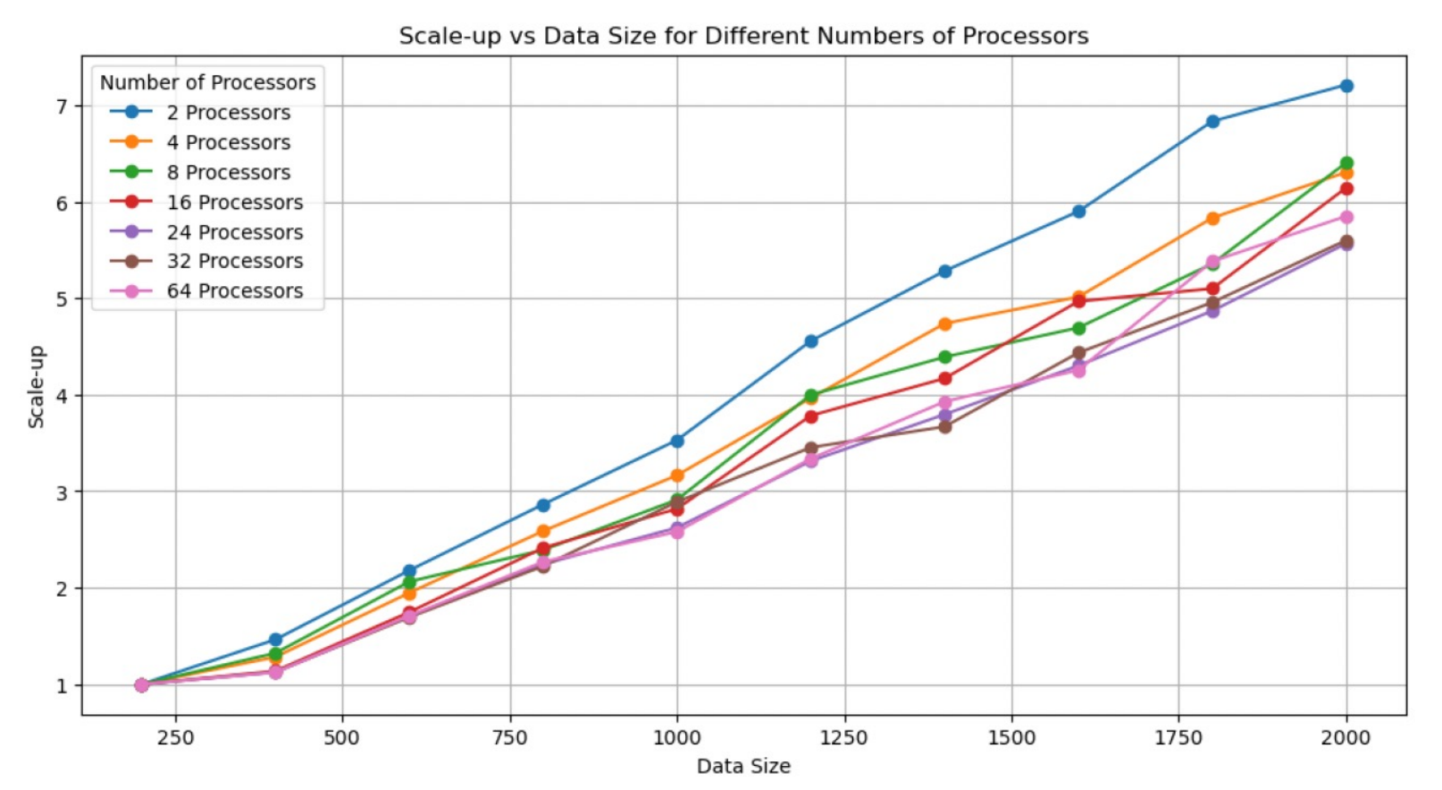

Figure 6: Scaleup Plot for Distributed Particles

Conclusion

The experiment evaluated the performance of the bat algorithm when parallelized using Apache Spark using 2 strategies: parallelizing data and parallelizing particles. The results indicate that both strategies exhibited initial efficiency gains with increasing cores used when running the Spark job, showing promising processing times, and demonstrated plateauing beyond 24 cores for both implementations, resulting in diminishing returns due to efficiency limitations.

The scaleup analysis demonstrated similar results where both implementations showed initial promising results in maintaining relatively consistent execution times for smaller datasets, while displaying increasing execution times for larger datasets, revealing limitations for larger datasets.

Overall, the study highlights the potential use of parallelization to enhance the performance of Particle Swarm Optimization techniques while demonstrating the need for optimizing overhead and resource management strategies to improve the performance for large- scale datasets to ensure more consistent and scalable parallel processing capabilities.

Acknowledgements

The authors acknowledge the support of the NDSU Department of Computer Science sponsored by NSF Directorate for Engineering and EPSCoR Co-Funding through NSF grant EEC2050175. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] X.-S. Yang, "Nature-Inspired Algorithms in Optimization: Introduction, Hybridization and Insights," *arXiv.org*, 2023. https://arxiv.org/abs/2401.00976#:~:text=Nature%2Dinspired%20algorithms%20are%20a (accessed Jul. 24, 2024).

[2] "Kennedy, J. and Eberhart, R. (1995) Particle Swarm Optimization. Proceedings of the IEEE International Conference on Neural Networks, 4, 1942-1948. - References - Scientific Research Publishing," *www.scirp.org*. https://www.scirp.org/reference/referencespapers?referenceid=1847917 (accessed Jul. 24, 2024).

[3] X.-S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," *arXiv:1004.4170 [physics]*, Apr. 2010, Available: https://arxiv.org/abs/1004.4170

[4] N. A. of S. Medicine Engineering, and, I. of Medicine, B. on H. C. Services, and C. on D. E. in H. Care, *Improving Diagnosis in Health Care*. National Academies Press, 2015. Accessed: Jul. 24, 2024. [Online]. Available: https://books.google.cl/books?hl=en&lr=&id=9vu9DwAAQBAJ&oi=fnd&pg=PR1&dq=example+healthcare

[5] "Towards an Integrative Big Data Analysis Framework for Data-Driven Risk Management in Industry 4.0," *ieeexplore.ieee.org*. https://ieeexplore.ieee.org/abstract/document/7427814

[6] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms," *Applied Soft Computing*, vol. 18, pp. 261–276, May 2014, doi: https://doi.org/10.1016/j.asoc.2013.09.018.

[7] M. G. Omran, A. P. Engelbrecht, and A. Salman, "IMAGE CLASSIFICATION USING PARTICLE SWARM OPTIMIZATION," *Advances in natural computation*, pp. 347–365, Aug. 2004, doi: https://doi.org/10.1142/9789812561794_0019.

[8] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on Apache Spark," *International Journal of Data Science and Analytics*, vol. 1, no. 3–4, pp. 145–164, Oct. 2016, doi: https://doi.org/10.1007/s41060-016-0027-9.

[9] H. Karau and R. Warren, *High Performance Spark: Best Practices for Scaling and Optimizing Apache Spark*. "O'Reilly Media, Inc.," 2017. Accessed: Jul. 24, 2024. [Online]. Available: https://books.google.cl/books?hl=en&lr=&id=90glDwAAQBAJ&oi=fnd&pg=PP1&dq=apache+spark+&ots=FB4RN-Xixd&sig=U1cmaDYpXa6l_dEa44fW0pgGiI8&redir_esc=y#v=onepage&q=apache%20spark&f=false