A Distributed Ant Colony Optimization in Edge Detection

Min Chen (chenm@newpaltz.edu)
State University of New York at New Paltz

Research Article

Keywords: Ant Colony Optimization, MapReduce, Spark

Posted Date: December 14th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-1153553/v1

Noname manuscript No.
(will be inserted by the editor)

A Distributed Ant Colony Optimization in Edge Detection

Min Chen

Received: date / Accepted: date

Abstract With the rise of image data and increased complexity of tasks in edge detection, conventional artificial intelligence techniques have been severely impacted. To be able to solve even greater problems of the future, learning algorithms must maintain high speed and accuracy through economical means. Traditional edge detection approaches cannot detect edges in images in a timely manner due to memory and computational time constraints. In this work, a novel parallelized ant colony optimization technique in a distributed framework provided by the Hadoop/Map-Reduce infrastructure is proposed to improve the edge detection capabilities. Moreover, a filtering technique is applied to reduce the noisy background of images to achieve significant improvement in the accuracy of edge detection. Close examinations of the implementation of the proposed algorithm are discussed and demonstrated through experiments. Results reveal high classification accuracy and significant improvements in speedup, scaleup and sizeup compared to the standard algorithms.

Keywords Ant Colony Optimization · MapReduce · Spark

1 Introduction

Edges are a basic image feature and they are present between a target and a background and between two targets, two regions, or two primitives. Most of an images information is carried in the edges. Usually, an image edge is a set of pixels around which the gray-level values exhibit a step change. Thus, edge detection technology is based on the discontinuity or mutation of gray

M. Chen
State University of New York
New Paltz
NY, USA
E-mail: chenm@newpaltz.edu

levels or textural characteristics between an object and its background. Edge detection is one of the most frequently used techniques in computer vision since the early 1970s [1]. Edge detection is a fundamental preprocessing step for a variety of application, including object tracking [2], segmentation [4], and active contours [3]. The image edges are detected and grouped into contours or surfaces to represent the boundaries of the objects. Edge detection aims to segment an image by finding and placing sharp discontinuities in gray level images. Many edge detection techniques have been reported in previous works such as Canny [5], Sobel [6], Prewitt [7], Log [8] etc. However, without optimization techniques, most of the existing detection techniques use a huge search space [9] for the image edge detection. Hence, the edge detection task consumes considerable memory and time.

Ant colony optimization (ACO) [10] is an evolutionary computation-based algorithm inspired by the natural phenomenon of ants' foraging. In this optimization method, ants follow an intelligent way of information transfer about the food sources to the rest of the colony with the help of some natural chemical secretion (pheromone) [9]. The deposited pheromone slowly evaporates over time while the ants are walking to and from the food source. Other ants observe the presence of pheromone trails and follow paths with highest concentration of pheromone. The shortest path receives the greatest pheromone concentration as more ants travel over it. Thus, the ants eventually will travel the shorter path with this positive feedback mechanism. Several techniques have been proposed to handle the edge detection using ACO. In [9] proposed an ACO edge detection using F ratio-based technique. An ant colony optimization based mechanism to compensate broken edges was presented in citeLu. It adopted four moving policies to reduce the computation load. An ACO-based edge detection approach was introduced in [15] to tackle the image edge detection problem. All these approaches indicate that ACO is a promising technique for edge detection.

With the advent of the big data era, traditional edge detection technologies are facing problem of poor edge detection and long runtimes; thus, this technology needs to be further analyzed and studied. It is computationally expensive using conventional approaches to edge detection because each set of operations is conducted for each pixel. As the size of image increase, the computation time increases quickly. In addition, because of the sheer volume and increasing complexity of image data being collected and created in the field of computer vision, the conventional approaches are inadequate to handle very large image data in a timely manner. In order to process this big data, both increased computer performance and improved algorithms that allow the exploration of large parameter spaces are necessary. MapReduce [13] is one of the most efficient big data solutions, which enables the processing of a massive volume of data in parallel with many low-end computing nodes. This programming paradigm is a scalable and fault-tolerant data processing tool that was developed to provide significant improvements in large-scale data-intensive applications in clusters.Edges are a basic image feature and they are present between a target and a background and between two targets, two

regions, or two primitives. Most of an images information is carried in the edges. Usually, an image edge is a set of pixels around which the gray-level values exhibit a step change. Thus, edge detection technology is based on the discontinuity or mutation of gray levels or textural characteristics between an object and its background.

This work proposes a novel algorithm in the field of edge detection in images using a MapReduce based ant colony optimization with filtering, to extract edge information from the pheromone matrix. The proposed algorithm takes advantage of both ACO and parallel computing. Th proposed algorithm is tested on a greyscale image set, a colored-greyscale image set and a color image set. The experimental results show that both statistically and visually this method performs better than the earlier reported techniques.

The rest of the paper is organized as follows. Section 2 reviews some related ACO work in literature. The proposed algorithm is demonstrated in Section 3. The experimental results and analysis are described in Section 4, and Section 5 concludes the paper.

2 Related Work

Different techniques of image edge detection have been presented in the literature. In [14], an ant colony optimization with enhanced equation of energy calculations for each state is proposed for edge detection. The edges of the images were extracted better than standard technique. An ACO-based edge detection approach is introduced to tackle the image edge detection problem in [15]. The study shows that the approach is able to establish a pheromone matrix that represents the edge information presented at each pixel position of the image, according to the movements of a number of ants which are dispatched to move on the image. Furthermore, the movements of these ants are driven by the local variation of the images intensity values. Superior performance of the proposed approach is provided in the experimental results.

In [16], a new algorithm for edge detection using ant colony search is proposed. The problem is represented by a directed graph in which nodes are the pixels of an image. Some modifications on original ant colony search algorithm (ACSA) are applied to adapt the problem. Algorithm parameters are determined with a large number of experiments. The results suggest the effectiveness of the proposed algorithm

In [17] and [18] , a swarm intelligence technique is applied in image feature extraction. An ant colony system is applied to build the perceptual graph of digital images. The edge feature in digital images is extracted based on the proposed machine vision model. The experimental results show that the ant colony system can effectively extract image features.

For edge detection through MapReduce, [19] proposed a parallel design and implementation model that runs on the Hadoop platform. The proposed parallel Otsu-Canny edge detection algorithm performs better than other traditional edge detection algorithms. The parallel approach reduced the running

time by approximately 67.2% on a Hadoop cluster architecture consisting of 5 nodes with a dataset of 60,000 images.

In [20], a MapReduce (MR) approach to perform edge detection of satellite images using a nature-inspired algorithm Artificial Bee Colony (ABC) is proposed. The study compares two edge detection approaches on Hadoop with respect to scalability parameters such as scaleup and speedup. The results suggest that the proposed algorithm scales comparatively better when compared to Canny Edge.

In [21] cloud services for high resolution video streams in order to perform line detection using Canny edge detection followed by Hough transform are proposed and evaluated. Both Canny edge detector and Hough transform algorithms in Hadoop and Spark are proposed. The experimental evaluation using Spark shows an excellent scalability and performance compared to Hadoop and standalone implementations for both Canny edge detection and Hough transform.

To summarize, research on ACO has been carried out from various dimensions. Improving the runtime efficiency of an ACO in edge detection still remains an open challenge. This motivates the design of a Mapreduce-based ACO, which is an efficient distributed ACO algorithm building on a highly scalable MapReduce implementation for edge detection. The aforementioned literatures demonstrate the effectiveness of parallel implementation of computer vision algorithms to achieve good scalability for real-world applications.

3 Proposed Approach

The proposed approach has three major steps: preprocessing step, map step and reduce step.

3.1 Preprocessing Step

The proposed algorithm starts with a preprocessing step. The images need to be preprocessed in order to convert them to computable formats which the map step can process. The images are converting into an image matrix pixel by pixel.

8-bit images use Eq. 1 and 24-bit images use Eq. 2, respectively.

$$a_2 = [x, y] \tag{1}$$

and for a color image, which has 24-bit color, a three dimensional array is created:

$$a_3 = [x, y, 3] \tag{2}$$

where x is the number of pixel columns in the input image, and y is the number of pixel rows in the input image. 3 indicates that the array is 3 dimensional image matrix.

3.2 Map Step

The next step of the proposed algorithm is the map step. This is the crucial part of the algorithm. Each preprocessed image matrix is the input of the map step. The image is segmented with a grid of 5-pixel × 5-pixel, 10-pixel × 10-pixel, 15-pixel × 15-pixel, 20-pixel × 20-pixel, and 25-pixel × 25-pixel, respectively. The ant colony algorithm uses the segmentation results to detect the images edges. The edges are emitted as the output of the map step. The detail explanation is listed in Algorithm 1.

Algorithm 1 Map Function.

```
Input: Image pixel data
Output: Key-value pairs for pixel pheromones
procedure MAPPER(input_data)
    for each line of input_data do
        Split line into components
        for range 0 to segment size as y do
            for range 0 to segment size as x do
                Image_matrix[x, y] = component
                Increment to next component from input_data line
    Process edge of image in image_matrix        ▷ The input_data is from the image
    for rows in edge as row do
        for pixels in row as pixel do
            print data from pixel
```

3.3 Reduce Step

The majority of processing is done in the mapper function. The main purpose of the reducer step is to condense the data back together. The output of detection results are stored in the HDFS as designed. The procedures are demonstrated in Algorithm 2.

3.4 Edge Detection with ACO

The proposed edge detection with ACO starts with the initialization of ants. It then runs iteratively to construct and update the pheromone matrix. To perform edge detection with ACO, two major functions need to be redesigned. First is how to determine the moving direction of each ant. Multiple observable conditions present at the time of movement along with a component of random chance need to be considered in a weighted approach. Eq. 3 is calculated to decide the moving direction of each ant.

Algorithm 2 Reduce Step.

Input: Sorted mapper output
Output: Reduced edge data

```
procedure REDUCER(sorted_input)
    current_word = NULL
    current_count = 0
    current_total = 0
    for lines in sorted_input as line do
        word,value = split(line)
        if current_word == word then
            current_count += 1
            current_total += value
        else                                    ▷ Found a new key in input
            if current_word != NULL then
                print data
            current_count = 1
            current_total = 0
            current_word = word
    if current_word == word then                ▷ Check for last line of input
        print data
```

$$w_{ij} = P[i,j]^{\alpha} + I[i,j]^{\beta} \tag{3}$$

where w_{ij} represents the neighboring pixel at location (i,j). $P[i,j]$ represents the current pheromone level at the pixel with the pheromone matrix P. $I[i,j]$ represents the pixel value of the original image. α and β represent the weights given to the pheromone level and pixel values respectively within this equation.

Another function to consider is the pheromone deposit. The equation that determines the pheromone drop quantity is (see Eq. 4).

$$d = \frac{|I[i,j] - I[x,y]|}{\lambda} \tag{4}$$

where d represents the pheromone drop amount after each ant step. I is the same as before. λ is the scaling factor of the pheromone deposit.

After every ant has completed a single step, the pheromone levels are then reduced to simulate evaporation found with ants in nature. This is to help reduce or remove pheromone deposits that may have been dropped in incorrect or less ideal locations. Hence, ants will be less likely to visit these undesirable locations. The equation for the pheromone evaporation is updated in Eq. 5.

$$P = \phi P \tag{5}$$

where ϕ is the pheromone evaporation factor. Once all steps are completed, the pheromone matrix P gets filled with different levels of pheromone at each pixel. To reduce the amount of excess edge data that could be caused by lower amount of pheromones, a cutoff level is used to select the useful edge data. The cut off set to be the mean value of the entire matrix. The values found that are lower than the cutoff will be ignored and the values above the cutoff will be retained. a filter method is adopted to select the cutoff level (see Algorithm 3). The detailed procedures of the proposed filtered ant colony optimization (FACO) are listed in Algorithm 4. The final edge results are produced for the following reduce step.

3.5 Filtered Ant Colony Optimization (FACO)

Noise in an image can effect an image visually for a person as well as a computer. Where a person will see a slightly less clear image, a computer will read extra inaccurate data. For this experiment, two variations of the edge detection algorithm were implemented and applied. These two approaches differ in how they determine pheromone drop quantities. The first is has already been covered and will now be refered to as the unfiltered approach. The second, which can be refered to as the filtered approach, will add additional processing to each pheromone drop made. This change will have a major effect by removing unwanted noise resulting in a more accurately produced edge. The filtered version operates as shown in Algorithm 3.

Algorithm 3 Filtering pheromone drop amounts.

Input: Current and previous pixels
Output: Pheromone drop quantity
procedure FILTER(*previous_pixel*, *current_pixel*)
  if percent-difference(*previous_pixel*, *current_pixel*) < limit **then**
    Drop no pheromones
  else
    Follow normal pheromone drop equation

Algorithm 4 Edge detection with FACO.

```
Input: Image
Output: The resulting edge
procedure FindEdges(image)
    Convert image to pixel data
    Create matrix of pixel data
    Initialize matrix of ants
    for all steps do
        for all ants do
            Calculate next step probabilities
            Choose direction to move in
            Deposit pheromones
        Simulate pheromone evaporation
    Calculate good pheromone level cutoff
    Filter good and bad pheromone edge data
    Create Edge from good pheromone edge data
```

4 Experiments and Results

4.1 Environmental Setup

The experiments are conducted both on a single-node Windows machine (Intel Core i7-7500U Kaby Lake (2.7 GHz, up to 3.5 GHz, 4 MB cache, 2 cores)) and m3.xlarge EC2 instances (Intel Xeon Processor E5-2670 v2 Ivy Bridge (2.50 GHz, 25MB Cache, 4 cores)) from AWS. The parameters values are listed in Tabel 1. 30 images [22] which can be categorized into greyscale, colored-greyscale and color are used for experimental purpose. The proposed parallel algorithm is run using the following distributed computing environment:

• Hadoop: The cloud compute cluster to assign namenode and datanodes, conduct work load balancing, resource scheduling and data replication.

• HDFS: The distributed file system that provides fault tolerance and high throughput access to large datasets.

• YARN: Provides job scheduling and resource management.

• SPARK: An extension of the MapReduce framework which makes use of Resilient Distributed Datasets (RDDs) as a fault tolerant data structure operated on in parallel.

4.2 Experimental Results with ACO in a Single Node Cluster

Detection results with the selective images of the proposed mapreduce based unfiltered ACO and the proposed FACO are illustrated in Fig. 1. As shown in the figure, the segmentation steps are 5, 10, 15, 20 and 25, respectively. The first image is a sample for the greyscale image set. The filtered edge contains less background noise as compared to the unfiltered edges on all segmentation

Table 1 Parameters and values used in the proposed algorithm.

Parameter	Value
α	0.1
β	10
ϕ	0.5
λ	10
ρ	0.22

steps. The second image is a sample of colored-greyscale image set. Similarly, it has less noise on the background and the shadows around the content of the image have been mostly removed as well. The third image is a sample of color image set. It is clear that the background noise has been removed as compared to the unfiltered approach.

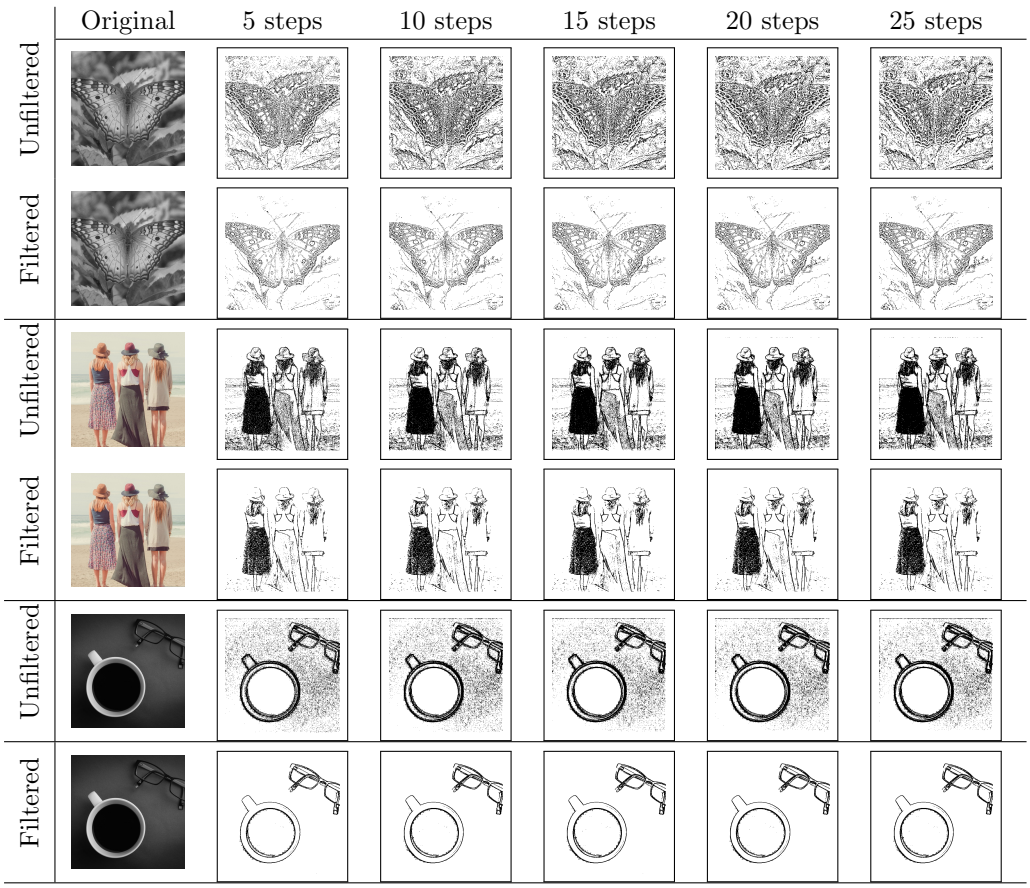

Fig. 1 Detection results using unfiltered ACO and filtered ACO.

4.3 Statistical Analysis

Mean Squared Error (MSE)(see Eq. 6) and Peak Signal-to-Noise Ratio(PSNR)(see Eq. 7) are used to compare the performance of the unfiltered ACO and the filtered ACO (FACO). Y is the vector of observed values and $\hat{Y}$ is the predicted value. MAX_I is the maximum possible pixel value of the image. It is set to 255 in this case.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{6}$$

$$PSNR = 20log_{10}(MAX_I) - 10log_{10}(MSE) \tag{7}$$

The result with the 25 segmentation steps of the whole image sets is listed in Table 2. The result indicates that the proposed FACO consistently performances better as compared to the unfiltered ACO in terms of MSE and PSNR. Moreover, the unpaired t-test with a significance level of 0.5 is performed on the data set of the two approaches. The null hypothesis is applied to verify the effectiveness of the proposed FACO. The P-values produced in Table 3 show a very promising value to conclude that the improvements of the filtered edges over the unfiltered edges is not merely coincidental.

4.4 Detection and Scalability Results using Multi-Node Clusters

The proposed FACO was later implemented and run on the Amazon web service to explore its scalability. The detection results of the selective images are listed in Fig. 2. The results are consist with the experiments conducted on a single-node cluster. The filtered ACO performs better than the unfiltered ACO in terms of background noise. Later, we conduct a scalability analysis whereby the segmentation step is increased by 5 dimensions for the whole image set. We observe a normal linear trend of the execution time for increasing segmentation step given that the date set is fixed in Fig. 3. The last experiment conducts a scalability analysis whereby the number of data nodes are increased linearly with the same image set. A fixed segmentation step with value 10 is selected. The result is illustrated in Fig. 4. The execution time of the proposed algorithm follows a linear trend with the data node range from 2 to 16. However, it doesn't perform linearly when the data node reaches 16 due to the throughput.

Table 2 25 step MSE & PSNR results in ACO and FACO, respectively.

	MSE		PSNR (dB)	
Image	**ACO**	**FACO**	**ACO**	**FACO**
bw_0_photo	21812.67	12919.88	4.74	7.02
bw_1_photo	33893.78	33363.99	2.83	2.90
bw_2_photo	28062.45	25991.47	3.65	3.98
bw_3_photo	32491.88	29575.38	3.01	3.42
bw_4_photo	39084.19	36740.86	2.21	2.48
bw_5_photo	22626.25	20167.21	4.58	5.08
bw_6_photo	34562.89	33989.78	2.74	2.82
bw_7_photo	28543.41	28417.90	3.58	3.59
bw_8_photo	24946.44	22346.87	4.16	4.64
bw_9_photo	24671.18	24325.25	4.21	4.27
colorbw_0_photo	30426.81	19747.05	3.30	5.18
colorbw_1_photo	33758.44	25942.12	2.85	3.99
colorbw_2_photo	32412.48	29974.06	3.02	3.36
colorbw_3_photo	32401.84	27957.81	3.03	3.67
colorbw_4_photo	47706.23	32408.86	1.35	3.02
colorbw_5_photo	26925.79	19558.21	3.83	5.22
colorbw_6_photo	45548.78	32258.15	1.55	3.04
colorbw_7_photo	27026.62	22210.96	3.81	4.67
colorbw_8_photo	52810.80	43495.22	0.90	1.75
colorbw_9_photo	19120.70	14257.94	5.32	6.59
color_0_photo	13707.68	10045.32	6.76	8.11
color_1_photo	28108.88	20801.43	3.64	4.95
color_2_photo	21972.93	12286.83	4.71	7.24
color_3_photo	16639.16	9879.53	5.92	8.18
color_4_photo	27000.44	16362.95	3.82	5.99
color_5_photo	11176.32	6173.68	7.65	10.23
color_6_photo	21781.89	15370.74	4.75	6.26
color_7_photo	20377.79	12744.96	5.04	7.08
color_8_photo	23624.78	14324.68	4.40	6.57
color_9_photo	42422.04	41814.26	1.85	-2.43

Table 3 T-test results.

Steps	**T-Value**	**P-Value**
5	-2.280	0.0262
10	-2.309	0.0244
15	-2.348	0.0222
20	-2.386	0.0202
25	-2.427	0.0183

Fig. 2 Greyscale AWS Edges.

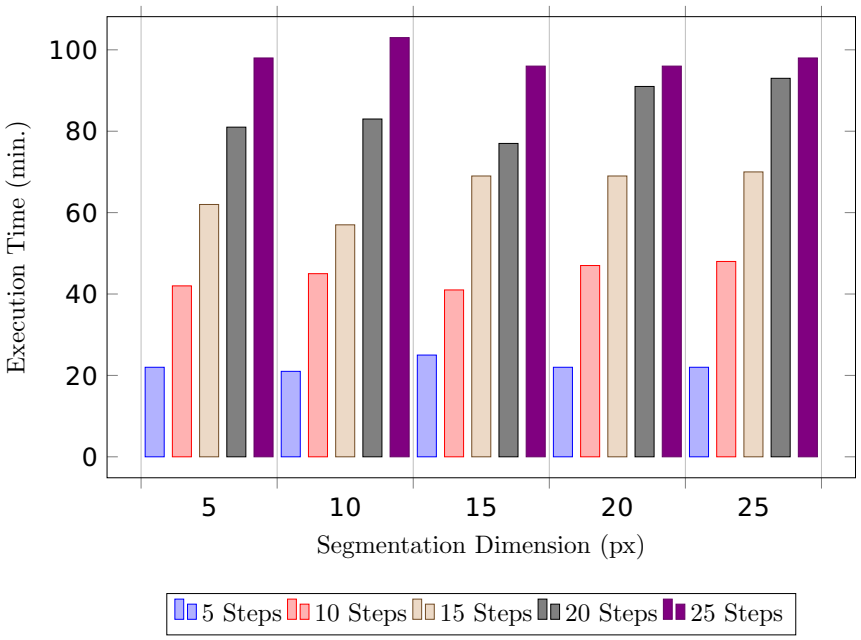

Fig. 3 Scalability of the proposed algorithm on EMR with 2 data nodes.

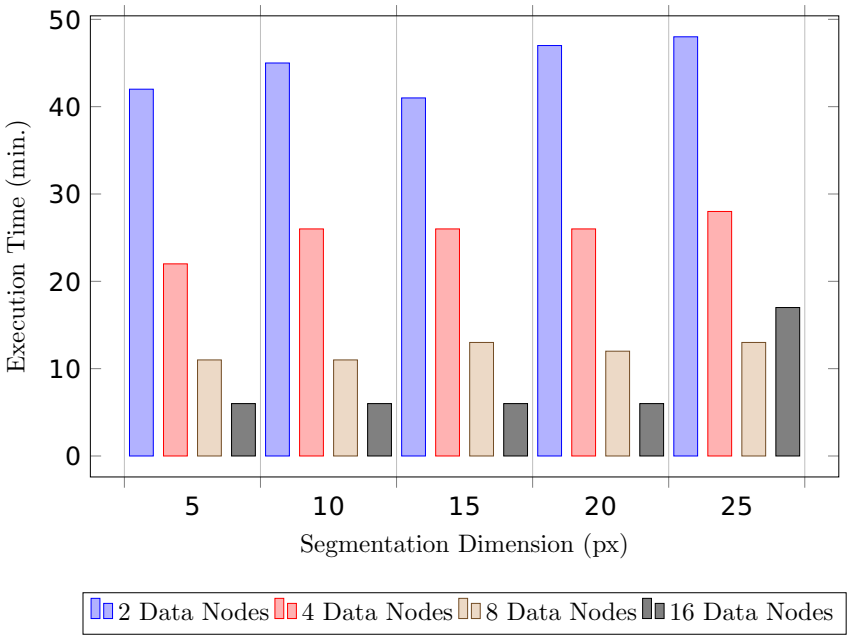

Fig. 4 Scalability of the proposed ACO with a segmentation step of 10.

5 Conclusion

With the rise of image data and increased complexity of tasks, the efficiency of conventional machine learning algorithms have been severely impacted by the nature of long training times and high computational cost. To be able to solve even greater problems of the future, conventional machine learning algorithms must maintain high speed and accuracy through economical means. MapReduce as one of the most efficient big data solutions enables the processing of a massive volume of images in parallel with many low-end computing nodes. This programming paradigm is a scalable and fault-tolerant data processing tool that was developed to provide significant improvements in large-scale data-intensive applications in clusters. In this paper, a novel ACO image edge detection with Hadoop MapReduce was successfully developed. The proposed FACO method takes advantage of the parallelism of ACO, its rapid convergence, and the virtue of the MapReduce paradigm. As shown in Section 4 the results demonstrate the effectiveness of applying ACO edge detection algorithm on the distributed system like Hadoop MapReduce. In the future, we propose to apply the algorithm in an image processing pipeline for object recognition and object tracking with improved speed and accuracy.

Abbreviations

AWS: Amazon web services
ACO: Ant Colony Optimization

Statements and Declarations

Ethics approval and consent to participate

The author Ethics approval and consent to participate.

Consent for publication

The authors consent for publication

Availability of data and materials

The data set is from Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

Competing interests

The author declares that she has no competing interests.

Funding

Not applicable.

Authors' contributions

The author has full contributions.

Acknowledgements

Not applicable.

Authors' information

Min Chen, Assistant Professor at State University of New York-New Paltz.

References

1. N. Senthilkumaran, R. Rajesh, Edge detection techniques for image segmentation a survey of soft computing approaches, International journal of recent trends in engineering, 1(2),(2009).
2. Yilmaz, Alper, Omar Javed, and Mubarak Shah. Object tracking: A survey. Acm computing surveys (CSUR) 38.4 (2006): 13.
3. Chan, Tony F., and Luminita A. Vese. Active contours without edges. IEEE Transactions on image processing 10.2 (2001): 266-277.
4. Chen, Min, and Simone A. Ludwig. Color Image Segmentation Using Fuzzy C-Regression Model. Advances in Fuzzy Systems 2017 (2017).
5. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal Mach Intell 8(6), 679698 (1986)
6. Gonzalez, R.C.,Woods, R.E.: Digital Image Processing. Addison-Wesley, MA (1992)
7. Maini, R., Sohal, J.S.: Performance evaluation of prewitt edge detector for noisy images. ICGST Int. J. Graph.Vis. Image Process. 6(3), 3946 (2006)
8. Marr, D., Hildreth, E.: Theory of edge detection. Proc. R. Soc. Lond. B 207(1167), 187217 (1980)
9. Ari, Samit, Dipak Kumar Ghosh, and Prashant Kumar Mohanty. Edge detection using ACO and F ratio" Signal, Image and Video Processing 8.4 (2014): 625-634.
10. A. Colorni, M. Dorigo and V. Maniezzo, Distributed optimization by ant colonies, Proceedings of ECAL 91, European Conference on Artificial Life, Elsevier Publishing, Amsterdam,1991.
11. Lu, De-Sian, and Chien-Chang Chen. Edge detection improvement by ant colony optimization. Pattern Recognition Letters 29.4 (2008): 416-425.
12. Tian, Jing, Weiyu Yu, and Shengli Xie. An ant colony optimization algorithm for image edge detection. Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on. IEEE, 2008.
13. Min Chen, Simone A. Ludwig and Keqin Li, Clustering in Big Data, Big Data: Management, Architecture, and Processing, (2017) 333-346.
14. Rezaee, A. (2008). Extracting edge of images with ant colony. JOURNAL OF ELECTRICAL ENGINEERING-BRATISLAVA-, 59(1), 57.

15. Tian, J., Yu, W., and Xie, S. (2008, June). An ant colony optimization algorithm for image edge detection. In 2008 IEEE congress on evolutionary computation (IEEE world congress on computational intelligence) (pp. 751-756). IEEE.
16. Nezamabadi-Pour, H., Saryazdi, S., and Rashedi, E. (2006). Edge detection using ant algorithms. Soft Computing, 10(7), 623-628.
17. Zhuang, X., and Mastorakis, N. E. (2005, April). Edge detection based on the collective intelligence of artificial swarms. In Proceedings of the 4th WSEAS International Conference on Electronic, Signal Processing, and Control.
18. Zhuang, X. (2004, July). Edge feature extraction in digital images with the ant colony system. In 2004 IEEE International Conference onComputational Intelligence for Measurement Systems and Applications, 2004. CIMSA. (pp. 133-136). IEEE.
19. Cao, J., Chen, L., Wang, M., and Tian, Y. (2018). Implementing a parallel image edge detection algorithm based on the Otsu-canny operator on the Hadoop platform. Computational intelligence and neuroscience, 2018.
20. Sharma, T., Shokeen, V., and Mathur, S. (2020). Distributed Approach to Process Satellite Image Edge Detection on Hadoop Using Artificial Bee Colony. International Journal of Service Science, Management, Engineering, and Technology (IJSSMET), 11(2), 80-94.
21. Iqbal, B., Iqbal, W., Khan, N., Mahmood, A., and Erradi, A. (2020). Canny edge detection and Hough transform for high resolution video streams using Hadoop and Spark. Cluster Computing, 23(1), 397-408.
22. Department of Computer Science and Artificial Intelligence at the University of Granada.