A Hybrid Ant Colony-Cellular Automata Algorithm for Improved Edge Detection of Images

Safia Djemame[1,2,*]

[1]*Computer Science Department, Faculty of Sciences, Ferhat Abbas University, Setif1, Setif, 19000, Algeria*
[2]*LRSD Laboratory, Ferhat Abbas University, Setif1, Setif, 19000, Algeria*

Abstract
The cellular automaton is an abstract model of computation. Investing the capabilities of cellular automata in image processing has proved to be promising. This paper presents a method for edge detection of images based on two dimensional cellular automata with extended Moore neighborhood model. This method uses ant colony optimization (ACO) to find the best linear rules of CA that produce satisfactory edge detection in one-time iteration. The performance of this approach is compared with some existing edge detection techniques. This comparison shows that the proposed method to be very promising for edge detection on various images with good quality and significantly low execution time.

Keywords
image processing, cellular automata, complex system, edge detection, ant colony optimization, metaheuristic, pheromone

1. Introduction

In the contemporary era, there is a wealth of data available, and a significant portion of this data comes in the form of images. These can include things like DNA microarrays, satellite maps, astronomy observations, and many others. Image processing has played a critical role in the technological advancements of recent years and has become a valuable tool in a variety of scientific research fields, including health, industry, and astronomy. This type of processing is primarily concerned with enabling machines to interpret and analyze images, extracting important information in the process. Among these preprossecing operations, segmentation represents a crucial process in image processing, and it involves various techniques, one of which is edge detection. One of the primary objectives of edge detection is to reduce the data size to be processed in subsequent stages of image processing. Although there are many edge detectors available in literature, none of them are perfect, and achieving optimal edge detection that produces high-quality edges while minimizing processing time is an area that requires ongoing research and development. The present work is a contribution to this ongoing quest for improved edge detection techniques. Many research works focus on finding increasingly efficient segmentation techniques to optimize the quality of contours and reduce computation time.In this study, our focus is on modeling an image with a powerful and complex system, the cellular automaton, which we combine with the metaheuristic ACO to extract transition rules that enable good contour segmentation. The present paper is organised as follows: In the first section, we introduce the working context and the domains of interest that we focus on. In the second section, we provide some fundamental concepts about cellular automata. In the third section, we explain the concept of ant colony algorithm. In the fourth section, we cite some related works. In the fifth section, we detail how we used the ACO algorithm to optimize a CA for the task of edge detection. In the sixth section, we explain the execution of ACO algorithm. In section seven, experimental results are shown, both visual and numerical, and a comparison is done. In section eight, we conclude with a summury and future prospects.

RIF 2024: The 13th International Conference on Research in Computing at Feminine, University of Abdelhamid Mehri, Constantine2, May 20–21, 2024
*Corresponding author.
safia.zazoua@univ-setif.dz (S. Djemame)
https://www.researchgate.net/profile/Djemame-Safia (S. Djemame)
0000-0002-8383-2819 (S. Djemame)

2. Basic Concepts of Cellular Automata

The popularity of cellular automata can be attributed to their simplicity and their ability to model complex systems, despite the fact that they are simple. Essentially, cellular automata can be considered a basic model of a decentralized extended system composed of numerous individual components, or cells. The interactions between these cells are limited to local interactions, and each cell has a specific state that evolves over time based on the states of its neighboring cells. Overall, this structure can be regarded as a parallel processing mechanism. Remarkably, even though the structure of cellular automata is straight-forward, when iterated numerous times, it can generate intricate patterns that have the potential to simulate a plethora of complex natural phenomena. A cellular automaton is a dynamic system comprising of a grid pattern, with each cell within this pattern being assigned a state from a finite pool of states. The system evolves in discrete time steps, with the state of a cell at time 't+1' being dependent on the states of a limited number of neighboring cells, as dictated by a specified transition rule. Every iteration of the system operates through the application of the same set of rules to every cell in the grid, thus generating a new 'generation' of cells entirely based on the prior one [1].

3. Ant Colony Optimization

In the early 1990's, Marco Dorigo and his team introduced the pioneering Ant Colony Optimization (ACO) algorithm, inspired by the behavior of ant colonies [2]. Ants are social insects that operate with the objective of advancing the survival of their colonies instead of focusing solely on the survival of individual ants. The ACO algorithm takes inspiration from the pervasive foraging behavior exhibited by ants to achieve their colony's goals. At the crux of their behavior lies indirect communication among the ants via chemical pheromone trails, which function to guide them toward the shortest paths between food sources and their nest. When ants forage for food, they initially move in random directions, leaving a trail of chemical pheromone behind. The pheromones attract other ants who, in turn, select paths with strong pheromone concentrations. Once an ant finds food, it evaluates its quality and quantity before carrying it back to the nest, leaving pheromones behind that reflect the quality and quantity of the food found. These pheromone trails attract other ants towards the food source, thus enabling the ants to discover the shortest paths between their nest and the food source. The success of ACO algorithms lies in their ability to operate efficiently on large problem instances with multiple solutions that require optimization. By exploiting the characteristics of real ant colonies, ACO algorithms offer unique and effective solutions to various discrete optimization problems. The collective behavior exhibited by real ant colonies has inspired the development of swarm intelligence techniques, such as the successful strand of ACO algorithms. These algorithms mimic the behavior of real ant colonies in their search for optimal solutions to discrete optimization problems. The utilization of real ant colony characteristics has enabled the development of effective and efficient multi-agent systems. Nonetheless, further research may be required to address the challenges of ACO algorithms to achieve optimal solutions for complex optimization problems.

3.1. The ACO algorithm

The general steps of ACO algorithm are as follows [2] :
* On each iteration or epoch, all ants build a path to the destiny node (the food source). For the next node selection, the probabilistic equation (1) is used:

$$P_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)}{\sum_{l \in N_i^k} (\tau_{il}^{\alpha}(t)} \, if j \in N_i^k \\ 0 otherwise \end{cases} \tag{1}$$

$P_{ij}^{k}(t)$ is the probability that ant k at point i moves to point j at time t.
N_i^k is the set of points that ant k can move to, starting from point i.

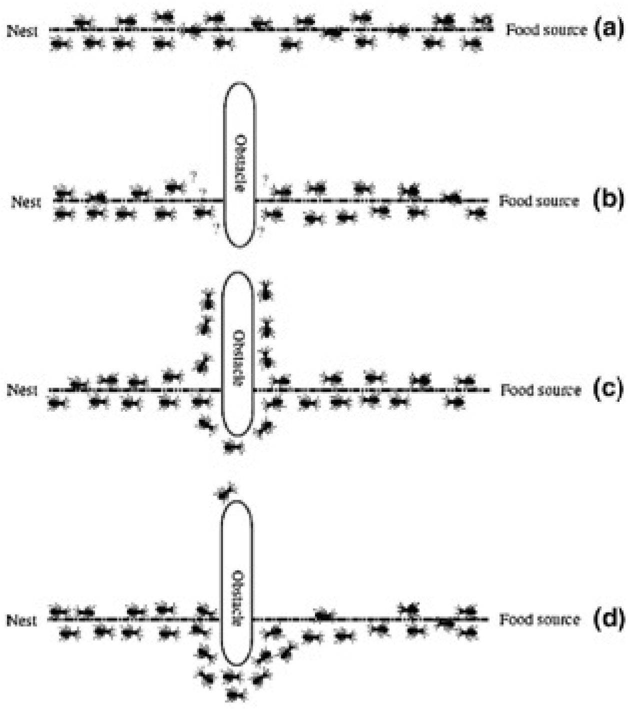


Figure 1: a)Path followed by ants, (b)Ants encounter an obstruction, (c)Ants find paths around the obstruction, (d)Ants follow the short path around the obstacle.[3]

$\tau_{ij}^{\alpha}(t)$ is the pheromone sensed in the pass that can lead from point i to point j, with α jump distance.
• Compute pheromone evaporation using equation (2):

$$\tau_{ij}(t) \leftarrow (1-p)\tau_{ij}(t) \tag{2}$$

p is the evaporation rate value of the pheromone trail. The evaporation is added to the algorithm in order to force the exploration of the ants and avoid premature convergence to suboptimal solutions. For p=1, the search is completely random.
• Update pheromone concentration using equation (3), allowing ants to add pheromone after each iteration of the program in the ant colony system:

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \tag{3}$$

m is the number of ants.
Informally, an ACO algorithm can be imagined as the interplay of three procedures [17]: ConstructAntsSolutions, UpdatePheromones, and DaemonActions.
• ConstructAntsSolutions manages a colony of ants that concurrently and asynchronously visit adjacent states of the considered problem by moving through neighbor nodes of the problem's construction graph. They move by applying a stochastic local decision policy that makes use of pheromone trails and heuristic information. In this way, ants incrementally build solutions to the optimization problem. Once an ant has built a solution, or while the solution is being built, the ant evaluates the partial solution that will be used by the UpdatePheromones procedure to decide how much pheromone to deposit.
• UpdatePheromones is the process by which the pheromone trails are modified. The trails value can either increase, as ants deposit pheromone on the components or connections they use, or decrease, due to pheromone evaporation. From a practical point of view, the deposit of new pheromone increases the probability that those connections that were either used by many ants or that were used by at least one ant and which produced a very good solution will be used again by future ants. Differently, pheromone evaporation implements a useful form of forgetting: it avoids a too rapid convergence of the algorithm

toward a suboptimal region, therefore favoring the exploration of new areas of the search space.
• Finally, the DaemonActions procedure is used to implement centralized actions which cannot be performed by single ants. Examples of daemon actions are the activation of a local optimization procedure, or the collection of global information that can be used to decide whether it is useful or not to deposit additional pheromone to bias the search process from a nonlocal perspective. As a practical example, the daemon can observe the path found by each ant in the colony and select one or a few ants (e.g., those that built the best solutions in the algorithm iteration) which are then allowed to deposit additional pheromone on the components/connections they used.

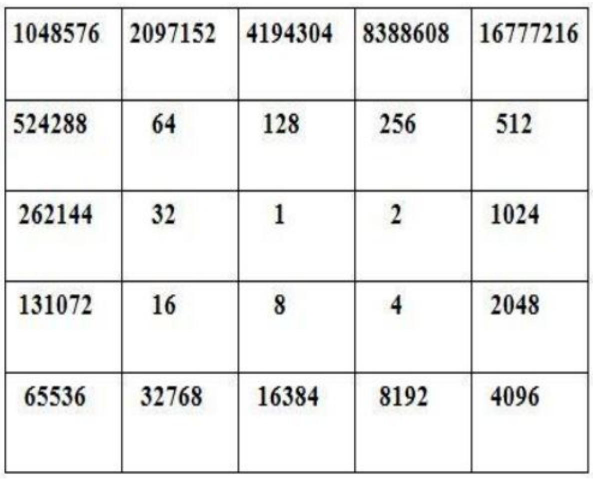
4. Related works

ACO was first proposed as a solution to the travelling salesman problem (TSP) [2], [4]. Since then, it has been successfully applied to numerous discrete optimization problems. Initially, it was used to tackle classical optimization issues, such as graph coloring, quadratic assignment [5], the maximum clique, vehicle routing [6], temperature controller [7] and scheduling [9]. More recently, ACO has been applied to diverse optimization scenarios, such as intelligent scheduling, design of communication networks, bioinformatics, circuit design-related cell placement, and machine learning[8]. In addition, some researchers have explored the applicability of ACO algorithms towards dynamic or stochastic problems, as well as towards solving multi-objective optimization problems[10], and real-world engineering optimization challenges [11]. In the literature, we found a few research articles that discuss the combination of ant colony algorithms and cellular automata: In [12], the authors propose an ACO-based approach to optimize the parameters of a cellular automata-based traffic simulation model. The algorithm is applied to a real-world traffic simulation problem, and the results demonstrate significant improvements in the model's accuracy. The article [13] presents a hybrid approach combining ACO algorithms and cellular automata to improve the optimization of traffic flow in urban areas. The proposed approach is evaluated on a simulated traffic network and compared to traditional traffic control methods, demonstrating superior performance in terms of reducing congestion and improving traffic flow. In the paper[14] the authors propose a hybrid approach combining cellular automata and ACO algorithms to address the dynamic network traffic assignment problem. The proposed approach is evaluated on a collection of real-world urban traffic scenarios, demonstrating superior performance compared to other optimization approaches. These papers showcase the effectiveness of combining ant colony optimization algorithms and cellular automata for a variety of optimization challenges, highlighting advancements in transportation simulation, urban traffic control, and dynamic network traffic assignment. In the present work, we hope to merge cellular automata with ACO approach for image processing, specifically edge detection. Cellular automata's computational talents and capabilities, along with ACO's metaheuristic ability, have promising possibilities for generating high-quality outputs.

5. The proposed approach

Our CA has 25 neighbor cells (extended Moore neighborhood), so the number of possible neighbor configurations is $2^{25} = 33554432$. The possible number of CA that can be conceived using combination of these configurations is $2^{33554432}$. The problem is to find the rules that perform edge detection the best within this large number of CA. A choice is established to only use CA with linear rules (only one neighbor configuration rule for each CA) which is beneficial in terms of computational performance of the CA and the substantial reduction in the search space to 33554432. The rule convention used to designate rules is shown in figure (2).

The number within each box represents the rule number associated with a neighbor configuration that only has that particular neighbor. So, if the next state of a cell depends only on its present state, it is represented as Rule 1. Similarly, if the next state of a cell is dependent only on its bottom neighbor, then it is represented as Rule 8 and so on. These twenty-five rules are known as fundamental/basic rules. All linear rules are derived using these basic rules which are expressed as the sum of the basic



1048576	2097152	4194304	8388608	16777216
524288	64	128	256	512
262144	32	1	2	1024
131072	16	8	4	2048
65536	32768	16384	8192	4096

Figure 2: The rule convention model

rules. For example, Rule 130 can be expressed as follows: Rule 130 = Rule 128 $\oplus$ Rule 2
ACO is used to optimize our search for CA rules that perform better edge detection. The idea is having a 5×5 space and ants occupy cells of this space and move around while generating pheromones in discrete time iterations. Pheromones value on every cell might increase if ants generate pheromones and decrease on each iteration by evaporation.

5.1. Parameters

The number of ants, the pheromone evaporation rate, and the randomness parameter are experimentation parameters of our implementation of ACO.

5.2. Initialization

This process creates the population of ants and put them in random positions of the 5×5 cells space. It also initiates the pheromone values in every cell to zero.

5.3. Convert position to rule number

The reason for using a 5×5 cells space in ACO is to project the ants to the CA neighbor model. The occupied cells in the ACO cells space define the current explored CA rule, an example of this is illustrated in figure (3), on the left the ants cells space with eight ants, on the right the CA neighbor model having fundamental rules in circle of the corresponding neighbors. The CA rule number in this example is 8446.
Quantifying the quality of edge produced by CA rule represents the amount of pheromone generated

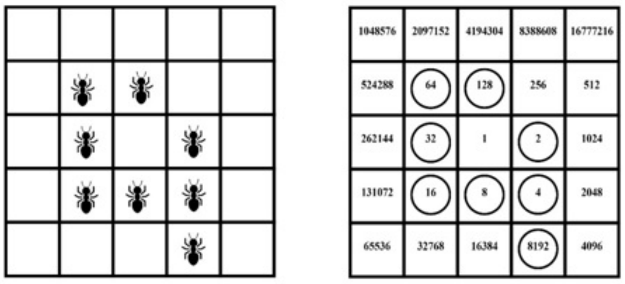


Figure 3: The translation of ants positions to a CA rule

by the ants. Thus, at each iteration all the ants generate the same amount of pheromone at their position. Multiple ants occupying the same cell are considered as one ant.

5.4. Update Pheromones

In this step the pheromone values increase in the cells occupied by at least one ant. The value increase is the same in these cells no matter how many ants in the cell. The increase in pheromones depends on the computed fitness.
In order to prevent the ACO from getting in local optimums, where pheromone values keep increasing in only select cells, a mechanism is set in place to push the optimization to explore more rules, by letting the ants generate pheromones only when significant fitness values are discovered. This is implemented by creating MinFitness variable, which is initiated by zero (0), and keeping track of the best fitness value (BestFitness). Whenever a discovered rule has greater fitness value than the best fitness, the BestFitness is updated using equation (4):

$$\text{BestFitness}_{t+1} = \begin{cases} \text{BestFitness}_t & \text{if fitness} \leq \text{BestFitness}_t \\ \text{fitness} & \text{otherwise} \end{cases} \tag{4}$$

The pheromone values are updated in all the occupied cells by the equation (5):

$$\text{Phero}(x, y)_{t+1} = \text{Phero}(x, y)_t + \max(0, \text{fitness} - \text{MinFitness}) \tag{5}$$

5.5. Update Ants

In every time iteration, an ant can move from the cell it occupies to any cell within a two-cell radius (horizontally, vertically or diagonally) without going out of the moving space. Figure (4) illustrates an example of an ant in the moving space, the cells in gray represent the possible next position of this ant in one-time iteration. The ant probably moves to the cell with the

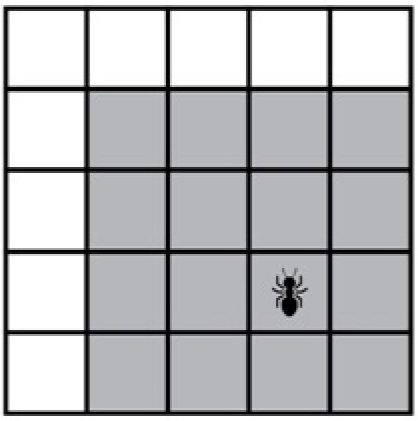


Figure 4: The ant moving space with one ant and its possible moves

highest pheromone value. To implement this probabilistic behavior, a randomness parameter R is introduced in the algorithm. The tendency for the ant to move to a cell x is evaluated using equation (6):

$$T(i) = Phero(i) + R * MaxPhero * Random() \tag{6}$$

T(i) is the tendency or probability that the ant moves to the cell i. Phero(i) is pheromone value at the cell i. R is the randomness parameter. MaxPhero is the maximum pheromone value of the cells that the ant can move to.Random() is a function that returns random values in the range [0,1[. The algorithm evaluates the values of T for all possible cells, and moves the ant to the cell with the highest value. Bigger values of the randomness parameter R imply higher probability that the ants move in a random way, while smaller values imply higher probability of movements to cells with greater pheromone values. R=0 means no randomness in the ants' movement.

5.6. Evaporate pheromones

The pheromone values in every cell are updated using equation (7):

$$Phero(x,y)_{t+1} = (1-p) * Phero(x,y)_t \tag{7}$$

p is the evaporation rate.

6. Execution of ACO algorithm

The initial parameters are set to: twenty-five (25) ants, evaporation rate of 5%, randomness parameter of 5. 500 cycles of the ACO algorithm have been run. After 300 iterations, we found that the best rules started to emerge. Some of the best rules identified in Moore neighborhood model (r=1) are: 153, 46, 394, 226, 184, 47, 395, some others in extended Moore neighborhood model (r=2): 262271, 1343, 4194767, 262643, 16637.

6.1. The dataset

The testing dataset consisted of a collection of images from the Berkeley University database that had well-defined ground truth edges [18]. The dataset comprised images captured in indoor and outdoor settings, encompassing a broad variety of edge types and complexities. This diverse collection of images enabled us to draw pertinent insights from the data, supporting the validity and usefulness of our findings.

6.2. Fitness fuctions

In order to evaluate the quality of edges produced by a CA rule, we need a function that measures how close are the results produced by this CA to the ground truth. Irrespective of the chosen optimization technique, the outcome greatly depends on the objective function employed, as it plays a critical role in determining the final results. As part of our study, we utilized two fitness functions, namely the Structural Similarity Index (SSIM) and Root-Mean-Square Error (RMSE), to assess the quality difference between the resulting image from cellular automata and the reference image. When dealing with images containing numerous intensity values, the RMSE can serve as a direct measure for assessing the similarity between the input and target image. However, it is widely recognized that the RMSE metric has certain limitations, as it fails to account for inter-pixel relationships, and may inadequately capture perceptual similarity. SSIM measures the image similarity taking into account three independent channels including luminance, contrast and structure [15]. It is the well suited measure for gray level images. The SSIM metric between two images x and y is defined by equation (8) :

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{8}$$

where $\mu_x, \mu_y, \sigma_x^2, \sigma_y^2, \sigma_{xy}$ are the mean of x, the mean of y, the variance of x, the variance of y, and the covariance of x and y, respectively. Following [16], C_1 is set to $(0.01 * 255)^2$ and $C_2 = (0.03 * 255)^2$. The RMSE is calculated according to equation (9):

$$RMSE = \sqrt{\frac{1}{MN} \sum_{r=0}^{M-1} \sum_{c=0}^{N-1} [E(r,c) - O(r,c)]^2} \tag{9}$$

where $O(r,c)$ is the original image (in our case, the ground-truth image) and $E(r,c)$ is the reconstructed image.

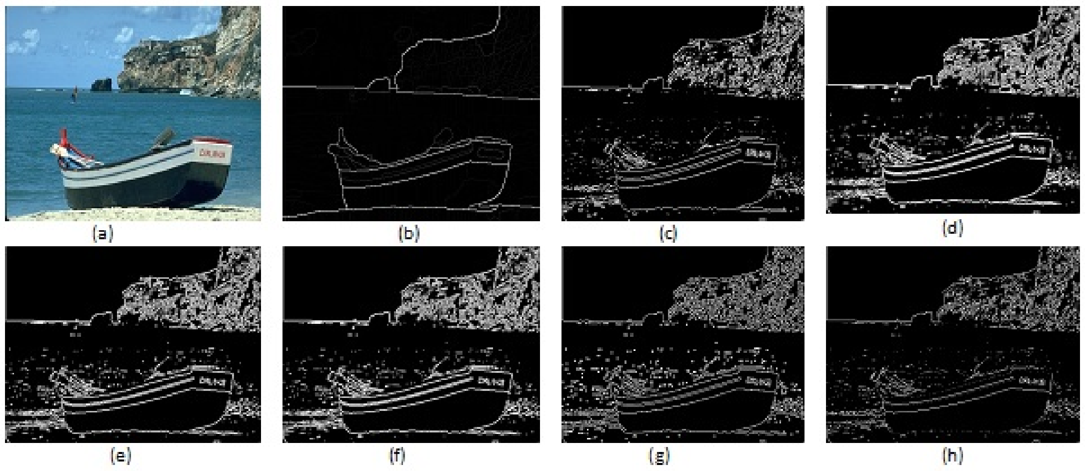


Figure 5: Edge detection on Boat image. (a) Original image (b) Ground truth (c) Rule 1343 (d) Rule 16637 (e) Rule 262271 (f) Rule 262643 (g) Laplacian (h) Scharr

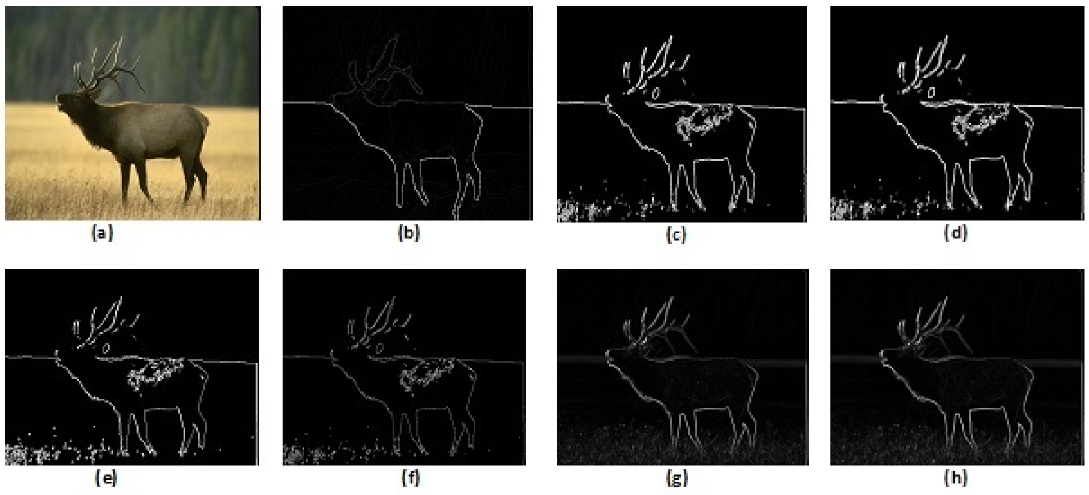


Figure 6: Edge detection on Reindeer image. (a) Original image (b) Ground truth (c) Rule 1343 (d) Rule 16637(e) Rule 262271 (f) Rule 262643 (g) Roberts (h) Canny

7. Experimental Results

In this section, we present the results of application of our hybrid ACO-CA algorithm to a set of images from Berkeley dataset. Among the numerous results collected, we have chosen to showcase three images. We demonstrate the execution of the rules 1343, 16637, 262271 and 262643, on the Boat, Reindeer and Castle images. It is essential to highlight that once the optimal rules have been identified, they can be directly applied to an image, thereby expediting the process and delivering the intended output efficiently. The findings presented in figures (5), (6) and (7), convincingly suggest that rules 1343, 16637, 262271, and 262643, obtained via the hybrid ACO-CA algorithm, yield favorable outcomes when compared to established methodologies such as Canny, Roberts, Laplacian, and Scharr. The edges generated through these rules exhibit continuity, finesse, and are precisely one-pixel wide, while the external contours are accurate, smooth and free of any noise. Additionally, these rules offer edges that possess a high level of detail. Our observations suggest that the results yielded by rules 16637, 262271 and 262643 are quite similar, while the contour obtained through Canny, Roberts, Laplacian and Scharr edge detectors are thinner, less pronounced, and less precise.

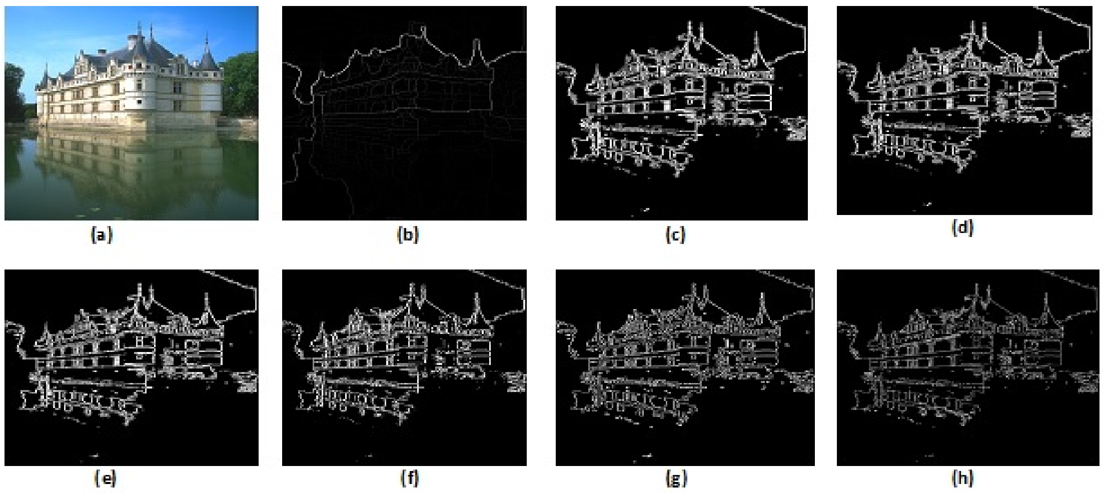


Figure 7: Edge detection on Castle image. (a) Original image (b) Ground truth (c) Rule 1343 (d) Rule 16637(e) Rule 262271 (f) Rule 262643 (g) Roberts (h) Canny

7.1. Fitness Values

Table (1) presents the optimal fitness values attained for the boat, reindeer, and castle images, after being subjected to evaluation using the RMSE and SSIM fitness functions. The results gathered from the tests are highlighted in table(1). It is evident from the numerical outcomes that rules 1343, 16637, 262271, and 262643 exhibit superior performance, delivering excellent fitness values. These results reinforce the visual outcomes produced by the aforementioned rules.

Table 1
Fitness Values for Images: Boat, Reindeer, and Castle

	Rule 1343		**Rule 16637**		**Rule 262271**		**Rule 262643**	
	RMSE	***SSIM***	***RMSE***	***SSIM***	***RMSE***	***SSIM***	***RMSE***	***SSIM***
Boat	0.242	0.98	0.201	0.99	0.214	0.99	0.218	0.99
Reindeer	0.198	0.99	0.197	0.99	0.209	0.98	0.232	0.97
castle	0.210	0.99	0.212	0.99	0.251	0.98	0.284	0.97

7.2. Execution Time

We illustrate the real-time edge detection process of the proposed method and compare it with established algorithms like Canny, Laplacian, among others, to carry out a comprehensive assessment. Table (2) highlights the execution time for various methods, including the proposed one, on three different images used to compute the edge detection time. The most noticeable observation from table (2) is as follows:
- The run time for Laplacian, Roberts, and Scharr is relatively similar and considerably less than Canny edge detector, which face challenges in achieving a real-time response. This obstacle represents one of the downsides identified for this method in our study.
- The CA rules extracted provide the least time-consuming approach and demonstrate variation from one rule to another. This outcome can be considered one of the most significant advantages of this research.

Table 2
Execution time for images: Boat, Reindeer and Castle

	Boat	**Reindeer**	**Castle**
Roberts	0.04165248	0.09558476	0.04625441
Scharr	0.02812411	0.02745476	0.02654187
Laplacian	0.04125389	0.03652478	0.02998415
Canny	0.12755896	0.13654418	0.15789965
Rule 1343	0.02426531	**0.01362247**	0.01812589
Rule 16637	**0.01406481**	0.01654147	0.01729865
Rule 262271	0.01567119	0.01112574	**0.014009952**
Rule 262643	0.01412859	0.01512652	0.01438215

8. Conclusion

This research introduces a novel technique for edge detection implementation via cellular automata utilizing an extended Moore neighborhood and optimization of the process of rules extraction with ACO algorithm. Even though this approach requires a larger rule space, our work employs a metaheuristic to uncover the optimal set of rules providing the best possible results. This strategy enables the identification of a subset of rules capable of producing high-quality edges within a single iteration. Experimental analysis performed on multiple images shows that these rules yield image outputs with enhanced contrast levels, while effectively smoothing the object edges present in the image. Comparisons between the resultant edges and the ground truth edges, alongside established edge detection techniques such as Canny, Roberts, Laplacian, and Scharr, are made using SSIM and RMSE fitness values, which confirm the robust performance of our approach. Notably, our rules demonstrate faster runtime compared to the aforementioned techniques.
As a future outlook for this study, we aim to investigate other metaheuristics for optimizing the set of rules in the cellular automata, such as grey wolf algorithm, bee algorithm, dolphin algorithm, and their quantum version. Additionally, we plan to introduce new principles of Deep Learning and utilize their robust learning capabilities to accelerate the optimization process and facilitate the attainment of optimal edge detection in the shortest possible time.

Declaration on Generative AI

The author has not employed any Generative AI tools.

References

[1] E. Fredkin, Digital Machine: A Informational Process Based on Reversible Cellular Automata. Physica-D (45), 7, 254–270, 1990.

[2] M. Dorigo, L. M. Gambardella, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem," IEEE Transactions on Evolutionary Computation, 1(1), 53-66, 1997.

[3] T. Dusan. "Swarm intelligence systems for transportation engineering: Principles and applications." Transportation Research Part C-emerging Technologies 16 (2008): 651-667. DOI:10.1016/J.TRC.2008.03.002

[4] A. Colorni, M. Dorigo, V. Maniezzo, Distributed optimization by ant colonies, In Proceedings of ECAL'91 European Conference on Artificial Life, Elsevier Publishing, Amsterdam, The Netherlands, pp 134-142, 1991.

[5] Maniezzo V, Colorni A. "The Ant System applied to the quadratic assignment problem", IEEE Trans. Data Knowledge Engineering, 11(5), 769–78, 1999.

[6] B. Bullnheimer, R.F. Hartl, C. Strauss Applying the ant system to the vehicle routing problem, In: Voss S., Martello S., Osman I.H., Roucairol C. (eds.) Meta- Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer, Boston, pp 285-296, 1999.

[7] S. Katiyar, A. Mittal, A. Q. Ansari, T. K. Saxena, "Ant Colony Algorithm Based Adaptive PID Temperature Controller," Proc. 7th Int. Conf. on Trends in Industrial Measurements and Automation (TIMA 2011), CSIR, Chennai, January 2011.

[8] Y. Sun, S. Wang, Y. Shen, X. Li, A. T. Ernst, M. Kirley "Boosting Ant Colony Optimization via Solution Prediction and Machine Learning", Computers and Operations Research, vol. 143, p. 105769, 2022.

[9] W. Deng, J. Xu and H. Zhao, "An Improved Ant Colony Optimization Algorithm Based on Hybrid Strategies for Scheduling Problem," in IEEE Access, vol. 7, pp. 20281-20292, 2019.

[10] L. Yongbo, H. Soleimani, M. Zohal. "An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives." Journal of cleaner production 227: 1161-1172, 2019.

[11] O. Mahamed GH, S. Al-Sharhan. "Improved continuous ant colony optimization algorithms for real-world engineering optimization problems." Engineering Applications of Artificial Intelligence 85: 818-829, 2019.

[12] X. Yang, J. Han, and Y. He . "Ant colony optimization for cellular automata parameter determination in transportation simulation." Journal of Transportation Engineering, Vol. 132, No. 11, pp. 853-862, 2006.

[13] R.B. Anggraini and D. Arini. "Ant colony optimization and cellular automata for urban traffic control system." 2014 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC), pp. 96-100, 2014.

[14] T.Q. Dai, S. Yang, and C.F. Cai. "A cellular automaton-ant colony optimization hybrid approach to the dynamic network traffic assignment problem." Journal of Advanced Transportation, Vol. 44, No. 3, pp. 193-209, 2010.

[15] Z. Wang, E.P. Simoncelli, A.C. Bovic, “Multi-scale structural similarity for image quality assessment”, in Proc. 37th IEEE Asilomar Conf. On Signals, Systems and Computers, Pacific Grove, CA, 2002.

[16] Z. Wang, A.C. Bovik, H.R. Sheikh, E.P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612,2004.

[17] M.Dorigo, T. Stützle. ""Ant colony optimization: overview and recent advances". Springer International Publishing, 2019.

[18] P. Arbelaez, M. Maire, C. Fowlkes and J. Malik, "Contour Detection and Image Segmentation Resources," [Online]. Available: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/ resources.html