Journal of Systems Engineering and Electronics
Vol. 32, No. 5, October 2021, pp.1062 – 1070

Memristive network-based genetic algorithm and its application to image edge detection

YU Yongbin[1,*], YANG Chenyu[1], DENG Quanxin[1], NYIMA Tashi[2], LIANG Shouyi[3], and ZHOU Chen[1]

1. School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China; 2. School of Information Science and Technology, Tibet University, Lhasa 850000, China; 3. Department of Economics, Stony Brook University, New York 11794, U.S.

Abstract: This paper proposes a mem-computing model of memristive network-based genetic algorithm (MNGA) by building up the relationship between the memristive network (MN) and the genetic algorithm (GA), and a new edge detection algorithm where image pixels are defined as individuals of population. First, the computing model of MNGA is designed to perform mem-computing, which brings new possibility of the hardware implementation of GA. Secondly, MNGA-based edge detection integrating image filter and GA operator deployed by MN is proposed. Finally, simulation results demonstrate that the figure of merit (FoM) of our model is better than the latest memristor-based swarm intelligence. In summary, a new way is found to build proper matching of memristor to GA and aid image edge detection.

Keywords: memristive network (MN), genetic algorithm (GA), edge detection, mem-computing.

DOI: 10.23919/JSEE.2021.000091

1. Introduction

The memristor was postulated first by Chua in 1971 as the fourth basic element of electrical circuits [1,2]. Later, its prototype model was fabricated by HP Inc. in 2008 [3]. The memristor is a two-terminal resistive element with memory effects, where the state of the device depends on one or more internal state variables and can be modulated depending on the history of external stimulation [4–6]. The compact device structure and the ability to both store and process information at the same physical locations make memristors and memristive networks (MN) attractive candidates for neuromorphic computing and image engineering applications [7–9].

With the memristor being a unit to store and compute information, the MN is designed to perform mem-computing [8–10]. Mem-computing is defined as a novel computing paradigm based on the ability of memory elements such as memristors [11]. In nature, mem-computing is a brain-inspired architecture composed of interacting memory elements controlled by external signals. This paper focuses on the design of MNs to implement mem-computing in biological intelligence nature-inspired meta-heuristic algorithms.

Nature-inspired meta-heuristic algorithms [10–26] focus on optimization problems by mimicking biological or physical phenomena, which can be classified into human-based algorithms, physics-based algorithms, swarm-based algorithms [10–13], and evolutionary algorithms. As for these algorithms, evolutionary algorithms inspired by the natural evolution, especially the genetic algorithm (GA), is of importance to solve optimization problems. GA inspired by the genetic process of biological organism, was developed by Holland [15]. Recently, increasing attention to GA has been paid by researchers and efforts have been made to improve GA for wide applications [16–26]. Specially, an effective genetic first-order statistical image segmentation algorithm was proposed in [25], which is the research field that we focus on.

Image segmentation can be classified into the threshold-based methods, region growing methods, hybrid methods, and edge detection methods. It aims to separate the desired foreground object from background [25,27]. By algorithm implementation and analysis [28,29], we find that the process of detecting edges is very similar to the evolution in GA, where edges in images are seen as high-level fitness individuals in population. In light of this design, we have implemented a new image edge detection method based on MN-based GA (MNGA) to obtain

Manuscript received January 21, 2021.
*Corresponding author.
This work was supported by the National Natural Science Foundation of China (61550110248), the Sichuan Science and Technology Department project (2019YFG0190), and the University of Electronic Science and Technology of China project (H04W170186).

the edge information from gray images.

The motivation of this paper is to precisely compare this seemingly different mem-computing and GA and understand their analogies and differences. The traditional GA is a biological intelligence algorithm usually employed to save the optimization problem and studied with software simulating [30]. Recently, research about memristive elements is increasing sharply, which brings new possibilities of hardware implementation of traditional intelligent algorithms like GA. In our MNGA, the fitness can be easily computed and stored, which is a meaningful innovation and attempt compared with the traditional software one. After that, to test and verify the presented MNGA, an image edge detection algorithm is presented which treats the process of edge detection as an optimization problem.

References [10] and [11] both present the ant colony optimization (ACO) algorithm implemented by memristor. Reference [10] has applied it to image edge detection. Reference [11] has applied it to the traveling salesman problem (TSP). The comparison between the proposed method and the methods in [10] and [11] is shown in Table 1. The main contributions of this paper are summarized as follows.

Table 1 Comparison of the proposed method with [10] and [11]

Method	Algorithm	Application
[11]	ACO	TSP
[10]	ACO	Image edge detection
The proposed method	GA	Image edge detection

(i) GA is implemented by MN for the first time. Key points of emulating a GA using hardware are how to restore the information of individuals and how to compute them iteratively, which can be appropriately solved by MN.

(ii) Image edge detection deployed by MNGA is explored. By the means of giving higher fitness to edge pixels, the edges can be recognized with GA. Then, we limit the logic range of individual crossover to keep the edge information continuous and repress isolated noise. Finally, when the GA is processing, more and more edges can be selected gradually, which will stop when the edges remain unchanged during two consecutive iterations.

(iii) The quality of detection is promoted compared with similar algorithms. The simulation results show that our algorithm works with higher quality and lower noise compared with the state-of-the-art ones.

The rest of this paper is organized as follows. Firstly, the design of the MNGA is presented in Section 2. In Section 3, we introduce the image edge detection algorithm based on MNGA. Finally, simulation results and comparison analysis are obtained in Section 4.

2. Computing model

2.1 GA

The GA premises that there are a certain number of individuals named “population” which perform biological behaviors in ways of natural selection, biomutation, and gene crossover. Every individual has its gene lists and processes fitness based on its gene lists. Fitness is the key to selection which eliminates low-level fitness individuals with high probabilities and high-level individuals with low probabilities. And with the evolution performing, more and more high-level individuals remain and the global fitness of the population can be promoted gradually. Finally, the optimal solution is obtained.

To simulate the GA, parameters need to be initialized as shown in Table 2. Fig. 1 shows a common GA process which can be implemented with selected $S_r(b)$, $C_r(b_1,b_2)$, and $B_r(b)$.

Table 2 Notations of GA

Symbol	Meaning
G	Generations
P	Population
B	Individual
C	Chromosome
gl	Gene list
$S_r(b)$	Rule of selection
$C_r(b_1,b_2)$	Rule of crossover
$B_r(b)$	Rule of biomutation
$FF(b)$	Fitness function

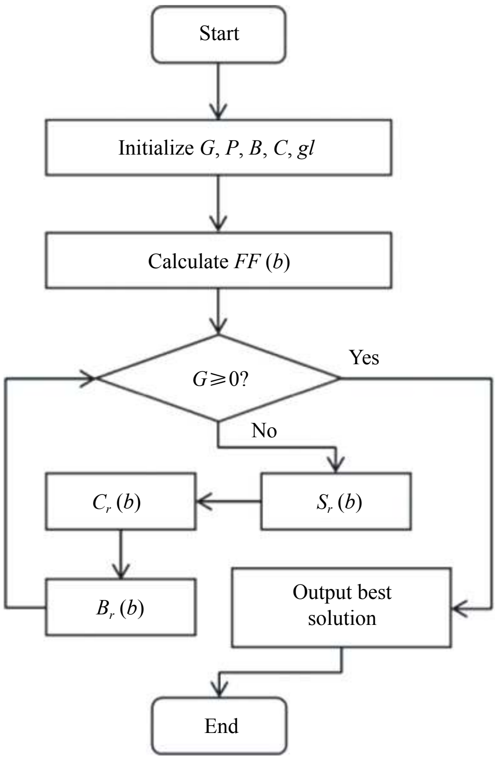


Fig. 1 Initialization and genetic operation of the GA process

Step 1 Initialize the MN and parameters of the GA, and the fitness of population can be calculated.

Step 2 If the algorithm is not terminated, $S_r(b)$ disposes individuals b one by one. If not, go to Step 6.

Step 3 After choosing two individuals b_1 and b_2 under the rule of S_r, $C_r(b_1,b_2)$ is completed to obtain a new individual.

Step 4 The process $B_r(b)$ will be implemented to decide whether individuals b are variable or not.

Step 5 After processes $S_r(b)$, $C_r(b_1,b_2)$, and $B_r(b)$, the new population is obtained. Then, the algorithm goes back to Step 2.

Step 6 The final population is the result of the GA.

2.2 MN

In this section, MN is discussed. An MN which includes basic elements (the memristor, the memcapacitor, the meminductor and their emulators) connecting grid points is a processor to solve the shortest path problems and build neural networks. Recent research [31] summarized that the type and functionality of memristive emulators are the key to network dynamics.

Inspired by [10] and [11], we find that there are still other traditional computing models which can be transformed with MN. Based on the structural, computing, and memory features of the MN, the MNGA is implemented to detect the edge information of images. If the memristive units like memristors are used to map the individuals of population, the MN is very suitable for GA which has a settled population. Then we also need a way to store the information of individuals for genetic processing. Finally, the MN can change the states of units expediently, which can map the changes of individuals naturally.

2.3 MNGA

The analogy and relation between the GA and the MN to solve optimization problems are then complete. To illustrate the mapping in Table 3, we assume an optimization problem $OP(x,y,z)$ where we find its maximum value with x, y, and z changing from 0 to 10. Firstly, every value group $[x,y,z]$ for $OP(x,y,z)$ is a feasible solution while that for the MN is a memristive subnet (which has three memristors in this case). A single memristance for the memristor maps a single value coding of x,y, or z for a feasible solution. Incidentally, in different GA models, the codings of the feasible solution are various such as binary coding, and floating coding, which require different memristor models. Finally, when $S_r(b)$, $C_r(b_1,b_2)$, and $B_r(b)$ are initialized, those approaches of solution in the GA maps the mem-computing approaches in Table 3. After this general discussion, we can now provide explicit applications based on MNGA.

Table 3 Mapping rules of GA and mem-computing

Nature	GA	Mem-computing
Population	Some feasible solutions	Memristive network
Individual	Feasible solution	Memristive subnet
Chromosome	Part of the feasible solution	Memristor
Gene	Feasible solution coding	Memristance
Natural selection	Solution selection	Memristor selection
Biomutation	Solution-coding change	Memristance change
Crossover	Addition of individual	Addition of memristor

The representation of chromosome by memristors is shown in Fig. 2. As we can see, there are six genes and each of them is coded by floating-point code. The memristors (denoted by $M_i, i=1,2,\cdots,6$) representation contains six genes and the value of each memristance is proportional to the corresponding gene. Furthermore, each memristor is connected to a current source which allows us to change the memristance independently and parallelly.

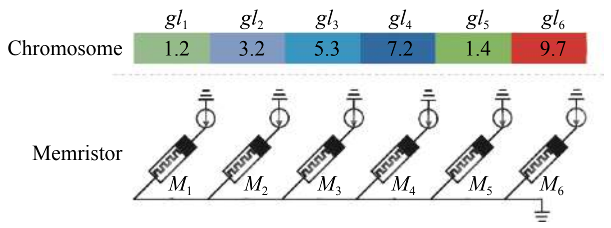


Fig. 2 Sample for implementing chromosome containing six genes by six memristors

The mapping operators of the chromosome and the memristive network are shown in Table 3. In the proposed method, the memristor is mainly used as a memory to read and wirte gene values. The selection operator is implemented by reading the memristance of memristors and the selection rule is run on memristances for selecting memristors. The biomutation operator can randomly select some memristors and change memristances randomly. Finally, the crossover operation is performed by reading and exchanging the memristance of the corresponding memristors. Compared with the traditional method, the parallel gene update of each operator can be realized by memristor deployment.

3. Edge detection based on MNGA

Detection of significant changes in an image is called edge detection. In this section, an MNGA-based edge detection algorithm is presented. Before introducing this algorithm, two design obstacles need to be considered. Firstly, in our early attempts, we find that the detection result is ineffective because of the noise in images. Secondly, it is difficult to design the mapping between an individual's fitness and an image's edge information. For the first obstacle, the filtering process which can suppress the white Gaussian noise is adopted. Furthermore, gray images are chosen for simulation to verify the feasibility of the algorithm. And for the second obstacle, we design a fitness function to quantitatively analyse every pixel, which means that the more similar one pixel is to the edge pixel, the higher-level fitness this pixel has. After solving these two obstacles, the MNGA based edge detection algorithm can be proposed.

The MNGA for image edge detection contains three steps. The first step is filtering. In order to properly detect the edges, there is a need to cancel local random variations caused by noise. The next step is called preprocessing which consists of computing the fitness of all pixels. The final step is edge detection where the new algorithm performs. In edge detection, the image is treated as the population and every pixel has their fitness which is calculated by $FF(x,y)$. Then an MN is used to express whether the corresponding pixel is an edge or not. With the GA performing, the low-level fitness pixels are eliminated while the corresponding memristor's memristance is installed to 0. Finally, at the end of the GA, the MN describes the edge information.

3.1 Filtering

To remove the impact caused by noise and promote the property of image edge detection, filtering is adopted.

3.2 GA operator based on memristive network

In this step, several preprocessing approaches will be done. Firstly, the 468×468 image Lena is presented (see Fig. 3), and the same sized MN needs to be completed where all memristors are two-value type memristors (for simplicity, memristance=0 or 1) to map the corresponding pixel. If memristance is 1, this pixel is an edge pixel, otherwise, it is not an edge pixel. Then all memristor's memristance is initialized to 1 which shows the corresponding pixel is edge, and the fitness of pixel is presented as follows:

$$FF(x,y)=\sum_{\substack{i=-1,0,1\\ j=-1,0,1}}|P(x,y)-P(x+i,y+j)| \qquad (1)$$

where $P(x,y)$ denotes the xth row and yth column pixel value in the image. $FF(x,y)$ denotes the fitness of pixel (x,y). We choose (1) which is simple to compute and has high performance. From (1), we know that the more different a pixel (x,y) is with its nearby pixels, the more higher $FF(x,y)$ is. According to this rule, it is easy to distinguish the edge pixels with the others.

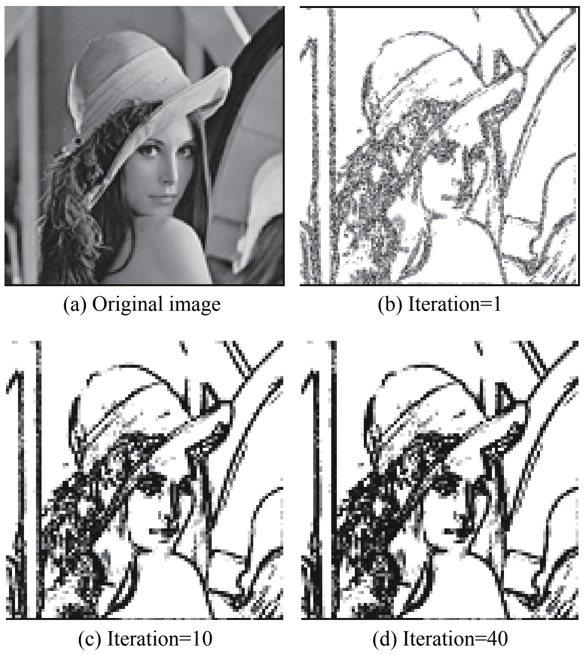


Fig. 3 Comparison of the quality of the edges detected from the Lena image

Remark Equation (1) calculates the difference in pixel values between a pixel and its neighbors. In general, the greater the difference is, the more likely it is to be an edge pixel. Spikes and noise are often isolated points eliminated by mutation during population iteration. In this case, more and more pixels remaining in the population are edges.

3.3 Edge detection

After the previous two processes, the model of the MNGA based edge detection can be presented as follows. The indispensable parameters and their meanings are shown in Table 4. The parameters need to be adjusted in Table 4 including R_c, R_b, F_m, C_c. All parameters have been adjusted several times to select the optimal ones. Furthermore, the selection processes of F_m and C_c are illustrated in Fig. 6 and Fig. 7 respectively.

Table 4 Parameters of MNGA-based edge detection

Symbol	Value	Meaning
R_c	0.4	Rate of crossover
R_b	0.05	Rate of biomutation
F_m	Mean value	Threshold value of fitness
$w\cdot h$	468×468	Resolution of image
$MN(x,y)$	0 or 1	Memristance
$P(x,y)$	$0-255$	Pixel value
$F(x,y)$	Calculated by fitness function	Fitness of $P(x,y)$ and $MN(x,y)$
C_c	4	The neighborhood size of crossover processing
Mean	33.667 8	Mean value of fitness

Step 1 Load the $m \times n$ pixels image to a matrix $\boldsymbol{P}(x,y)$, the pixel value is from 0 to 255, and prepare a same sized memristive network $MN(x,y)$ where memristors are two-valued and $MN(x,y)=0$ or 1. x and y denote the coordinate information and their ranges are $[1,m]$ and $[1,n]$.

Step 2 Randomly choose 50% memristors of the MN and set their memristance as 1 and the others as 0. Meanwhile the image is processed by the Gaussian-filter whose σ is 1.6.

Step 3 Count all pixels' fitness $FF(x,y)$. Set up parameters as shown in Table 2.

Step 4 Perform the selection process $S_r(b)$, where b is an elect pixel which will be eliminated if its fitness is lower than F_m.

Step 5 Perform the crossover processing $C_r(b_1,b_2)$, where b_1 and b_2 are two elect pixels and a new pixel will generate if the condition R_c is satisfied.

Step 6 Perform the biomutation processing $B_r(b)$, where b is an elect pixel.

Step 7 Perform Steps 4–6, if the number of generatons by Step 3 is larger than the number of eliminatons by Step 6.

Step 8 The MN becomes the edge information of the image, with 1 denoting the edge.

4. Simulation results

4.1 Experiment and analysis

By initializing parameters shown in Table 4, the simulation results are presented in Fig. 3 and Fig. 4. The edge information is picked out gradually. And in this case, the algorithm is convergent at the 39th generation, as shown in Fig. 4.

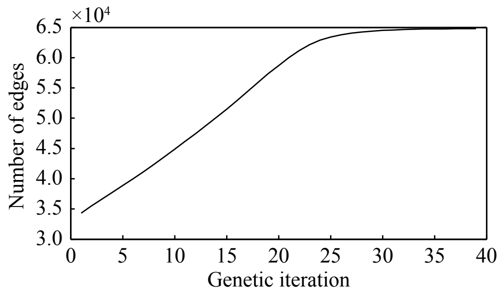


Fig. 4 Convergence rate of GA

In order to properly detect the edges, there is a need to cancel local random variations caused by noise. The detection result without median filtering is presented in Fig. 5 (c) where there is much residual noise compared with Fig. 5 (d). By comparing Fig. 5 (c) and Fig. 5(d), we can easily know that the filtering plays an important role in the detection algorithm at the same experimental conditions.

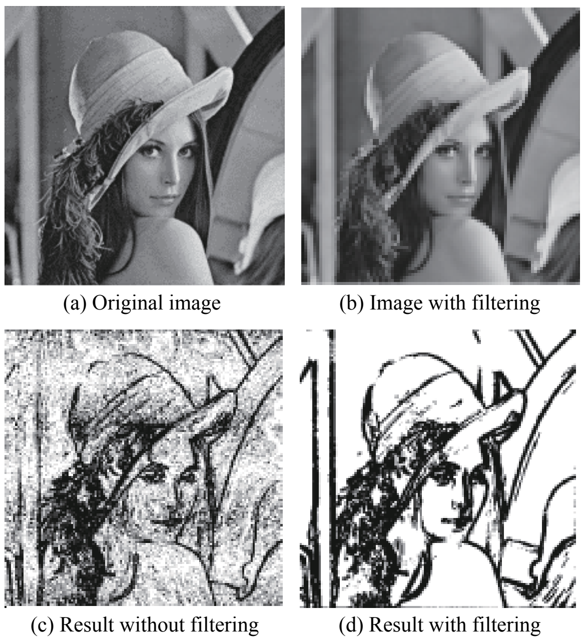


Fig. 5 Comparison of the edge detection quality with and without filtering conditions

In this case, the time complexity $T(n)$ of our detection algorithm can be obtained as follows:

$$T(n) = L \times O(T_{S_r}(n) + T_{C_r}(n) + T_{B_r}(n)) \tag{2}$$

where n denotes the number of pixels, $T_{S_r}(n)$ denotes the time complexity of $S_r(b)$, $T_{C_r}(n)$ denotes the time complexity of $C_r(n)$, $T_{B_r}(n)$ denotes the time complexity of $B_r(b)$, and L is iteration which is independent of n. Firstly, $S_r(b)$ will iterate over the population. Then the process of crossover is determined by R_c. Finally, R_b affects $T_{B_r}(n)$. Based on Table 4, the calculation of $T_{S_r}(n)$, $T_{C_r}(n)$, and $T_{B_r}(n)$ is provided as follows:

$$\begin{cases} T_{S_r}(n) = O(n) \\ T_{C_r}(n) = O[R_c \cdot (15n)] = O[0.4 \cdot (15n)] = O(n) \\ T_{B_r}(n) = O(R_b \cdot n) = O(0.05 \cdot n) = O(n) \end{cases} . \tag{3}$$

Consider (2) and (3) together and we get the time complexity $T(n)$ is equal to $O(n)$. And the space complexity $S(n)$ consists of the storage of image, the storage of fitness, and the storage of MN. After introducing the MNGA and the edge detection algorithm, we know that $S(n)$ is $O(n)$.

To compare the quality with different F_m, Fig. 6 (a) and Fig. 6 (b) show the experimental results which are compared with the results as shown in Fig. 3 (d). In Fig. 6 (a), the value of F_m is initialized to 0.5 mean, which retains more edges than Fig. 6 (b). With F_m increasing, fewer edges are selected but more noise filtered. Meanwhile, the detailed relationship between F_m and the number of edges is presented in Fig. 7. We know that the result in Fig. 3 (d) is a compromising choice to balance the discrete degree of edges and the number of details, which means our MNGA edge detection can perform various effects as differently required. To demonstrate this point, Fig. 7 and Fig. 8 vividly present the changes of the number of edges with different parameters. Three different values of F_m are employed to reflect the influence of F_m. And the number of edges is proportional to F_m. Three different values of C_c are employed to reflect the influence of C_c. And the number of edges is proportional to C_c. The other key parameter is C_c which defines a square window to limit the range of crossover between two individuals. If C_c is equal to 4, two individuals can reproduce a new individual only when they are in the window with a diameter of 4. Suitable size of C_c can effectively eliminate noise and retain serial edges, which can be proved in Fig. 8. With C_c increasing from 2 to 10, the number of edges is maximal when C_c is 4.

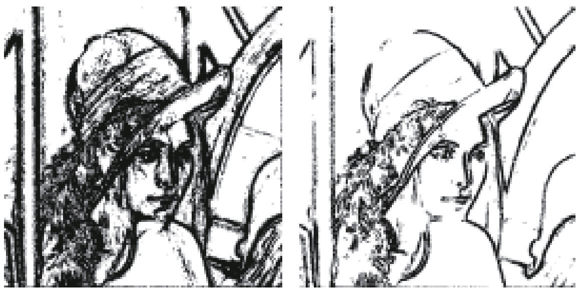


Fig. 6 Comparison of the quality of the edges detected from Lena image shown in Fig. 2 (a) with different parameters

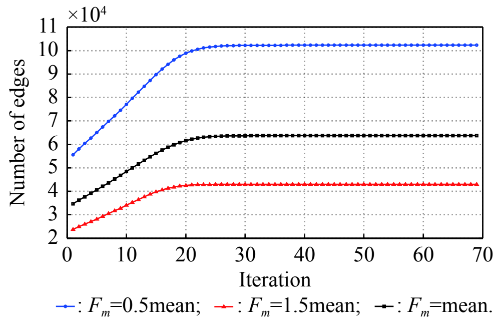


Fig. 7 Impact of F_m

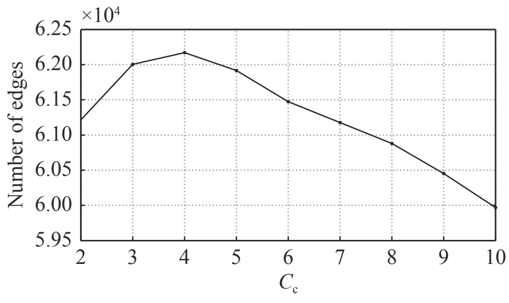


Fig. 8 Impact of C_c

4.2 Contrastive analysis

In this paragraph, a contrastive analysis is presented. Fig. 9 and Fig. 10 show the edge detection results, where Fig. 9 (a) and Fig. 10 (a) are the original images. Based on those two comparisons, we find that our MNGA can obtain more details than the algorithm in [10]. Meanwhile, in [10], there are many discrete edges which should be continuous originally. In our result, more continuous edges are retained than those in [10].

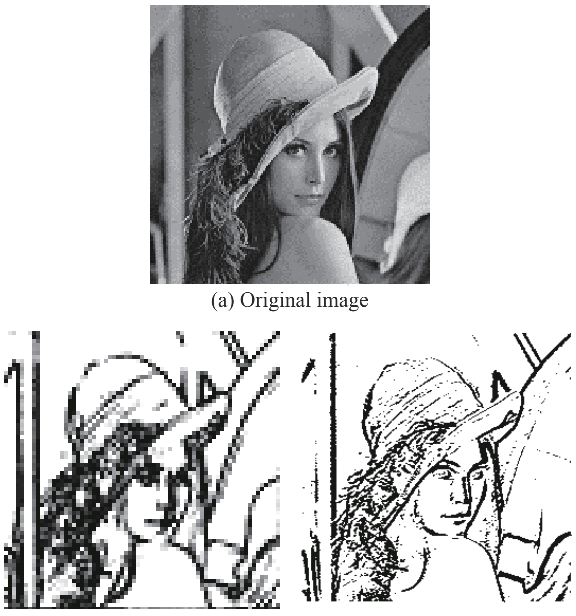


Fig. 9 Comparison of the Lena image edge detection with that in [10]

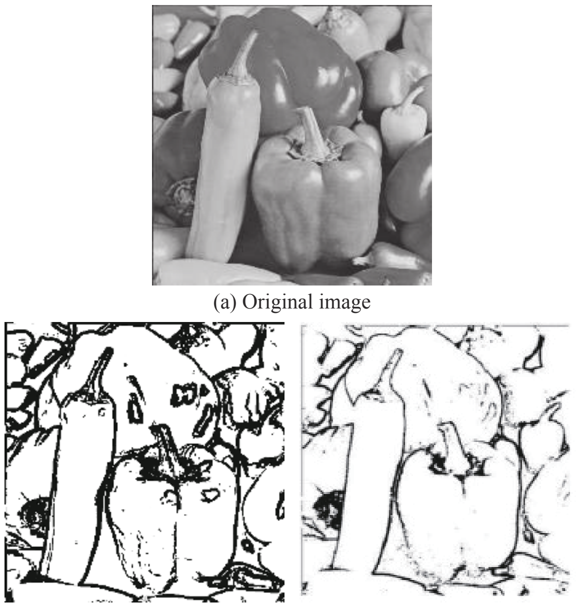
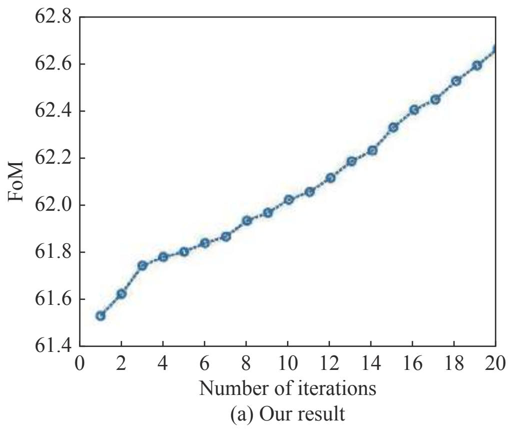


Fig. 10 Comparison of the Pepper image edge detection with that in [10]

Fig. 11 shows the FoM [32,33] based on the correlation between results obtained by the edge detector and the results obtained by the Canny algorithm. After only one iteration, our result is proved to be higher than the best result from [10].



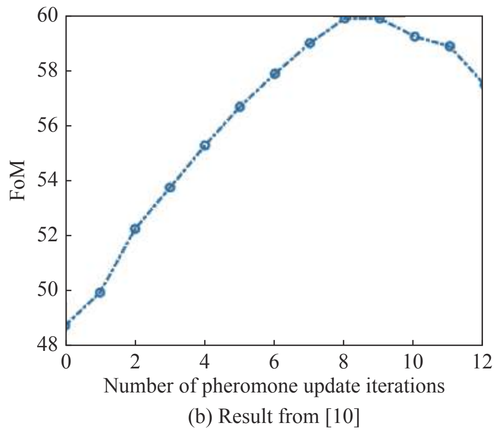
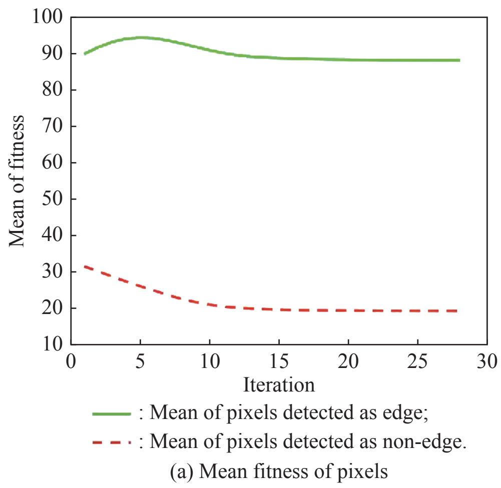


Fig. 11 Comparison of the FoM on Pepper image with that in [10]

Fig. 12 shows the change of the mean and the standard deviation of fitness with iteration. Firstly, it can be found that the mean and the standard deviation of edge pixels are both higher than that of the non-edge pixels. This shows that the fitness of the edge pixels is often higher than that of the non-edge pixels. However, because the standard deviation is large enough, the edge pixels are not concentrated on the high fitness process.



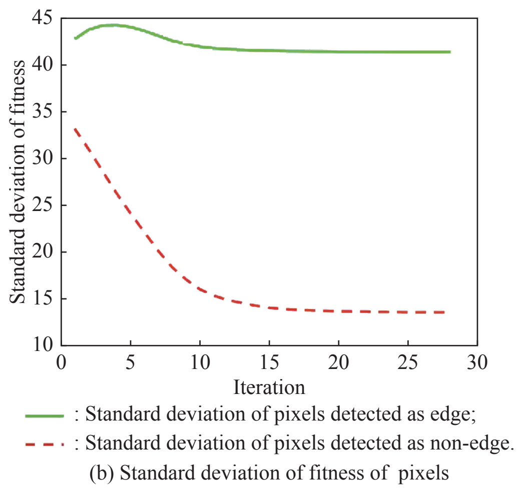


Fig. 12 Change of mean and standard deviation of fitness with iteration

5. Conclusions

In this paper, a new MNGA is proposed. In light of the computing and memory features of the MN, the relationship between the MN and the GA is built. After comparing the MN and the GA system structures, self-feedback, and storage of unit information, we find that the GA processes high suitability level with MN. With this mapping implemented, an edge detection algorithm is designed for images, where the pixels of images are treated as individuals of a population (also the units of the MN). Experimental results show the high quality of our algorithm compared with a similar computing model. Furthermore, we discuss some parameters of the MNGA which can impact the detection result, thus more experiments to confirm the most suitable parameters are required. This design is a novel point to explore the possibility of MN-based biological intelligence algorithms. There are still some others like GA that can be implemented in the similar way.

These preliminary results advocate further investigation of the proposed design method in the future. For the next step, we would like to compare more other well-known related algorithms regarding detection quality. Then, the adaptation and self-adaptation of those parameters studied in this paper should be worthy of study. Also, more other MN-based nature-inspired meta-heuristic algorithms can be designed to form a unified system.

References

[1] CHUA L O. Memristor—the missing circuit element. IEEE Trans. on Circuit Theory, 1971, 18(5): 507–519.

[2] CHUA L O, KANG S M. Memristive devices and systems. Proceedings of the IEEE, 1976, 64(2): 209–223.

[3] STRUKOV D B, SNIDER G S, STEWART D R, et al. The missing memristor found. Nature, 2008, 453(7191): 80–83.

[4] TOUR J M, HE T. Electronics: the fourth element. Nature, 2008, 453(7191): 42–43.

[5] CHUA L O. The fourth element. Proceedings of the IEEE, 2012, 100(6): 1920–1927.

[6] ADHIKARI S P, SAH M P, KIM H, et al. Three fingerprints of memristor. IEEE Trans. on Circuits and Systems I: Regular Papers, 2013, 60(11): 3008–3021.

[7] PREZIOSO M, MERRIKH-BAYAT F, HOSKINS B D, et al. Training and operation of an integrated neuromorphic network based on metal-oxide memristors. Nature , 2015, 521(7550): 61–64.

[8] DU C, CAI F, ZIDAN M A, et al. Reservoir computing using dynamic memristors for temporal information processing. Nature Communications, 2017, 8(1): 2204.

[9] SHERIDAN P M, CAI F, DU C, et al. Sparse coding with memristor networks. Nature Nanotechnology, 2017, 12: 784–789.

[10] PAJOUHI Z, ROY K. Image edge detection based on swarm intelligence using memristive networks. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2018, 37(9): 1774–1787.

[11] PERSHIN Y V, VENTRA M D. Memcomputing implementation of ant colony optimization. Neural Processing Letters, 2016, 44(1): 265–277.

[12] FISTER I, PERC M, KAMAL S M, et al. A review of chaos-based firefly algorithms: perspectives and research challenges. Applied Mathematics and Computation, 2015, 252: 155–165.

[13] FISTER I, PERC M, LJUBI K, et al. Particle swarm optimization for automatic creation of complex graphic characters. Chaos, Solitons & Fractals, 2015, 73: 29–35.

[14] MIRJALILI S, LEWIS A. The whale optimization algorithm. Advances in Engineering Software, 2016, 95: 51–67.

[15] HOLLAND J H. Adaptation in natural and artificial systems. Cambridge, U.S.: MIT Press, 1992.

[16] DEB K, PRATAP A, AGARWAL S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation, 2002, 6(2): 182–197.

[17] LIU J, TANG X. Evolutionary search for faces from line drawings. IEEE Trans. on Pattern Analysis & Machine Intelligence, 2005, 27(6): 861–872.

[18] DAS S, SUGANTHAN P N. Differential evolution: a survey of the state-of-the-art. IEEE Trans. on Evolutionary Computation, 2011, 15(1): 4–31.

[19] GHAMISI P, BENEDIKTSSON J A. Feature selection based on hybridization of genetic algorithm and particle swarm optimization. IEEE Geoscience and Remote Sensing Letters, 2015, 12(2): 309–313.

[20] XUE B, ZHANG M J, BROWNE W N, et al. A survey on evolutionary computation approaches to feature selection. IEEE Trans. on Evolutionary Computation, 2016, 20(4): 606–626.

[21] BIANCO S, CIOCCA G, SCHETTINI R. Combination of video change detection algorithms by genetic programming. IEEE Trans. on Evolutionary Computation, 2017, 21(6): 914–928.

[22] FALISZEWSKI P, SAWICKI J, SCHAEFER R, et al. Multi-winner voting in genetic algorithms. IEEE Intelligent Systems, 2017, 32(1): 40–48.

[23] NAGAMANI A N, NAYAK A S, NANDITHA N N, et al. A genetic algorithm based heuristic method for test set generation in reversible circuits. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2017, 37(2): 324–336.

[24] GONG D W, SUN J, ZHUANG M. A set-based genetic algorithm for interval many-objective optimization problems. IEEE Trans. on Evolutionary Computation, 2018, 22(1): 47–60.

[25] GAO B, LI X Q, WOO W L, et al. Physics-based image segmentation using first order statistical properties and genetic algorithm for inductive thermography imaging. IEEE Trans. on Image Processing, 2018, 27(5): 2160–2175.

[26] FISTER I, SUGANTHAN P N, JR I F, et al. Artificial neural network regression as a local search heuristic for ensemble strategies in differential evolution. Nonlinear Dynamics, 2016, 84(2): 895–914.

[27] GONZALEZ R C, WOODS R E. Digital image processing. Beijing: Publishing House of Electronic Industry, 2007.

[28] GUDMUNDSSON M, EL-KWAE E, KABUKA M. Edge detection in medical images using a genetic algorithm. IEEE Trans. on Medical Imaging, 1998, 17(3): 469–474.

[29] SAENTHON A, KAITWANIDVILAI S. Development of new edge-detection filter based on genetic algorithm: an application to a soldering joint inspection. International Journal of Advanced Manufacturing Technology, 2010, 46(9): 1009–1019.

[30] KAO Y T, ZAHARA E. A hybrid genetic algorithm and particle swarm optimization for multimodal functions. Applied Soft Computing, 2008, 8(2): 849–857.

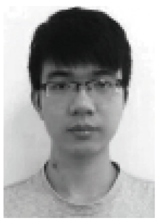
[31] VENTRA M D, PERSHIN Y V. The parallel approach. Nature Physics, 2013, 9(4): 200–202.

[32] PRATT W K. Digital image processing. New York: Wiley, 1978.

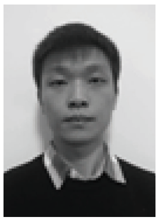
[33] YI S, LABATE D, EASLEY G R, et al. A shearlet approach to edge analysis and detection. IEEE Trans. on Image Processing, 2009, 18(5): 929.

Biographies

YU Yongbin was born in 1975. He received his Ph.D. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 2008. He is currently an associate professor at the School of Information and Software Engineering, UESTC. His research interests include big data and memristor.
E-mail: ybyu@uestc.edu.cn



YANG Chenyu was born in 1994. He received his B.S. and M.S. degrees from the UESTC, Chengdu, in 2016 and 2019 respectively. His research interests include memristor and genetic algorithm.
E-mail: chenyu_yang_divine@163.com



DENG Quanxin was born in 1996. He received his B.S. degree from the UESTC, Chengdu, in 2018. He is currently pursuing his M.S. degree with the School of Information and Software Engineering, UESTC. His research interests include memristor and genetic algorithm.
E-mail: 2507120731@qq.com

NYIMA Tashi was born in 1964. He received his Ph.D. degree from Sichuan University, Chengdu, in 2009. He is currently a professor in Tibet University. His research interests include computer network and information system.
E-mail: nmzx@tibet.edu.cn

LIANG Shouyi was born in 1993. He received his B.S. degree from Sichuan University, Chengdu, in 2016. He received his M.A. degree from the Stony Brook University, New York, in 2020. His research interests include memristor and econometrics.
E-mail: 15008231993@163.com

ZHOU Chen was born in 1998. She received her B.S. degree from the UESTC, Chengdu, in 2020. She is currently pursuing her M.S. degree with the School of Information and Software Engineering, UESTC. His research interests include memristor and genetic algorithm.
E-mail: 756349535@qq.com