TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 19, No. 1, February 2021, pp. 96~104
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v19i1.16418

Measuring memetic algorithm performance on image fingerprints dataset

Priati Assiroj[1], H. L. H. S. Warnars[2], E. Abdurrachman[3], A. I. Kistijantoro[4], A. Doucet[5]
[1,2,3]Computer Science Department Binus Graduate Program, Bina Nusantara University Jakarta, Indonesia
[1]Information System Department, Universitas Buana Perjuangan Karawang, Indonesia
[1]Politeknik Imigrasi, Ministry of Law and Human Right, Republic of Indonesia, Indonesia
[4]School of Electronics Engineering and Informatics, Institut Teknologi Bandung, Indonesia
[5]L3i Laboratory, La Rochelle University, France

Article Info

Article history:

Received Apr 17, 2020
Revised Jul 2, 2020
Accepted Aug 29, 2020

Keywords:

Biometrics
Fingerprints
Image
Memetic algorithm
Performance

ABSTRACT

Personal identification has become one of the most important terms in our society regarding access control, crime and forensic identification, banking and also computer system. The fingerprint is the most used biometric feature caused by its unique, universality and stability. The fingerprint is widely used as a security feature for forensic recognition, building access, automatic teller machine (ATM) authentication or payment. Fingerprint recognition could be grouped in two various forms, verification and identification. Verification compares one on one fingerprint data. Identification is matching input fingerprint with data that saved in the database. In this paper, we measure the performance of the memetic algorithm to process the image fingerprints dataset. Before we run this algorithm, we divide our fingerprints into four groups according to its characteristics and make 15 specimens of data, do four partial tests and at the last of work we measure all computation time.

Corresponding Author:

Priati Assiroj
Computer Science Department Binus Graduate Program
Bina Nusantara University
Jl. Raya Kebonjeruk, Anggrek, DKI Jakarta, Indonesia
Email: priati@binus.ac.id

1. INTRODUCTION

The increase of digital crime, such as data manipulation and image and signature falsification, and many various illegal transactions, is equal to the request of its solutions. Computer-based biometrics validations are important tools to improve the system's security. The most important of biometrics are measure physiology and behavior characteristics that allow making authentication to personal identity. Biometric authentication in the computer-based application is an important thing caused by the huge of sensitive data in the computer system and it increases rapidly.

Personal identification has become one of the most important terms in our society regarding access control, crime and forensic identification, banking, and also computer system [1]. Biometric features that can be used to identification include the iris, voice, DNA, and fingerprints. According to [2], the fingerprint is the most used biometric feature caused by its unique, universality, and stability. The fingerprint is widely used as a security feature for forensic recognition, building access, automatic teller machine (ATM) authentication, or payment. Automatic fingerprint identification has become an interesting research topic for two decades [3], in accordance with [4] fingerprint because of its unique, size, and peculiarity. Nowadays, to get a fingerprint

recognition tool is very easy, many corporations and organizations use it to identify their member [1]. Fingerprint recognition could be grouped into two various forms, verification [5] and identification [6]. Verification compares one on one fingerprint data. Identification is matching input fingerprint with data that saved in the database. Therefore, identification is an extension of verification that compare one fingerprint data to many, and note that identification is more complex than verification.

2. RESEARCH METHOD

2.1. Memetic algorithm

Memetic algorithm (MA) is an algorithm based on the Neo-Darwinian Evolution concept and Dawkins' opinion about memes as cultural evolution unit which can make improvements to itself. MA is a heuristic method which has similar characteristic to genetic algorithm (GA), combined with local search method, that can improve quality solutions [7]. On MA, local search is used for local improvement which can be applied before or after selection, crossover, and mutation process. Local search is also very useful to control search space. MA can achieve a better result than GA but need more computational time.

MA is a simple algorithm but flexible and powerful [8, 9], that can find quality solutions in many challenging problems [10-12]. Optimization problems will involve dozens of variables thus need good coding devices to overcome computational time. This algorithm will very useful in data mining process such as agglomeration and text analysis [13, 14], biomedical, such as DNA and molecular simulation [15-17], network problems [18, 19], feature selection technique [20], molecular simulation [16], forecasting [21], quantum chemistry [22], spectroscopy analysis [23], geophysics analysis [24], medicine invention [25], genome study [26], and many more.

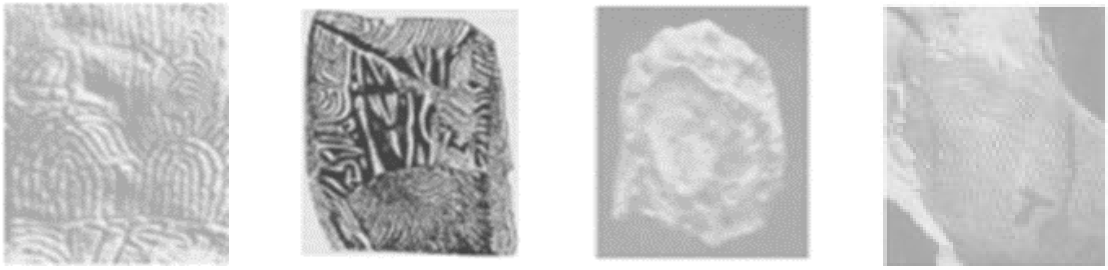
Specifically, the use of algorithms for image processing has been done by [27], they have used memetic algorithm for image brightness enhancement; [28] use this algorithm to detect face emotion, while [29] use this algorithm for letter recognition. Handwritten recognition conducted by [30] and [31], retina recognition by [32], sub-pixel mapping imagery by [33], image classification by [34], and fingerprint matching with the memetic algorithm by [35], [36] and [5].

2.2. Biometrics

Biometrics are special characteristics of human which is unique on every individual and can be a reference of identification and verification. Biometrics characteristics can be categorized into two types, physical and behavior. We all know that physical types such as fingerprint, iris, retina, and face, also behavior type such as voice, signature, gait, palm geometry, handwritten, electrocardiograph (ECG) [35]. The fingerprint is one of the most reliable biometrics features caused by its unique, and it is impossible to find the same fingerprint between two different people. This biometrics feature can automatically recognize someone by pattern recognition and determine authenticity physiology characteristics.

2.2. Fingerprints

The fingerprint is a graphic ridge and valley pattern at the tip of a human finger. Human fingerprints can be found on many historical objects as shown in Figure 1. This finding proves that ancient people were aware and gave special attention to fingerprint individuality, despite they did not have a scientific basis [37]. The history of fingerprint begins in 1684, morphologist from England, Nehemiah Grew, published his scientific paper about the ridge, groove, and pore structure of fingerprint [37]. In 1788 Mayer has described fingerprint in detail [38].



(a) (b) (c) (d)

Figure 1. Fingerprints on historical objects, (a) neolithic sculpture [38], (b) stonehenge at Goat Island [37], (c) clay cup from China 300 BC [37], (d) traces on the light of Palestinian 400 AD [38]

Thomas Bewick in 1809 uses the fingerprint as its characters and it is believed as the milestone of scientific fingerprint [38]. In 1823, Purkinje classifies fingerprint as the first. He was classified fingerprint to nine categories based on the ridge configuration, and in 1880 Henry Fauld and Herschel have done scientific fingerprint recognition [38]. This invention is the basis of modern fingerprint recognition. At the end of the nineteenth century, Sir Francis Galton researched more about fingerprints. Galton introduces a small feature for fingerprint matching in 1888. The most important progress in fingerprint recognition occurred in 1899 when Edward Henry generates "Henry System" [37]. So at the beginning of the twentieth century, the form of fingerprints can be understood well.

3. IMPLEMENTATION MODEL

There are several processes conducted to implement this algorithm. We start the process with reading folders and fingerprint files, we will do the local search process. Then convert fingerprint files to string array form and convert string array to binary code form with Base64, the process will be looped until the entire data converted successfully. The next process is selection or elitism for parent candidates from the total population then make it crossing-over to get new offspring. The last process is a mutation or clones the offspring. Figure 2 shows the flowchart of the process of the memetic algorithm, and a detailed process will be described in the next paragraph. In this paper we implement MA on 7200 synthetic image fingerprint dataset of FVC2006 with characteristics as shown in Table 1.

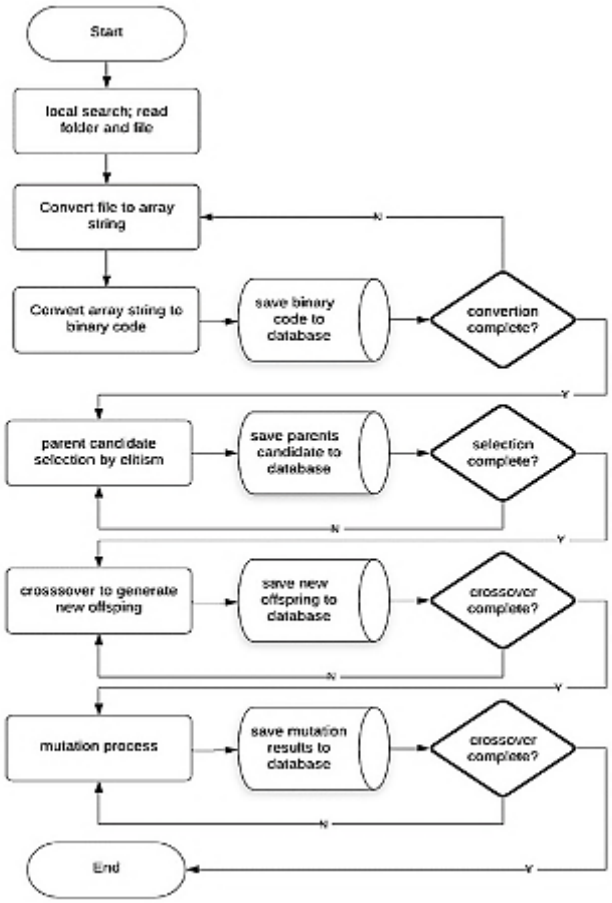


Figure 2. Flowchart of memetic algorithm

Table 1. Groups of fingerprint image data

Groups	Contents
A	100% fully-sized fingerprints image
B	60% fingerprints with dark color boundaries
C	60% fingerprints with bright color boundaries
D	80% fingerprints with bright color boundaries and unclear image

Then we conduct several steps as described below:

– Local search. We start the local search by reading the folder and fingerprint file h identification and data processing. We divide image fingerprint data into four groups, as shown in Table 1, based on its characteristics.
– Conversion. In this step is convert fingerprints image data to string array then convert string array to binary code use Base64. In this algorithm, we will do some swap, crossover, and mutation process, inside there, are crossing and data cloning that needs binary input and output, so we need it to be converted to binary code. We present the code in java language to convert string array to binary code as shown in Figure 3.
– Elitism. In this step, we conduct some selection steps. Theoretically, in elitism, we need a 20% random sampling of the population. In the preliminary experiment, we only use a 2% random sampling of the population caused by the limitation of our hard drive. Figure 4 shows the elitism code with java language. At the end of code, we finish with "null" to clear variables that are not used so that RAM is not full.
– Crossover. Crossover is a hybridization step of every selected sample to get new offspring. The selected samples divide into two groups, male and female, then they will be swapped one to the whole selected sample in another group, this process will be looped until the last selected sample. Figure 5 is the illustration of the crossover process in this paper and Figure 6 shown crossover code in java language.
– Mutation. This is the last process, where new offspring from the crossover process cloned then reversed several binary lines to get new offspring. Figure 7 shows the code of the mutation process in java language.

All steps were coded in java language with Netbeans IDE 8.2 and Experiments carried out use host and guest operating system in 1 PC with VMBox. This system runs in Intel i5-2540M with 16 GB of memory.

```
public void digFiling(String folderPathStr, String encodedString, String finalString) throws Exception {
    File folderPath = new File(folderPathStr);
    long waktu = 0;
    for (File fileEntry : folderPath.listFiles()) {
        String mimeType = Files.probeContentType(fileEntry.toPath());
        if (fileEntry.isDirectory()) {
            digFiling(fileEntry.getPath(), encodedString, finalString);
        } else {
            if (mimeType != null && mimeType.contains("bmp")) {
                long now = System.currentTimeMillis();
                byte[] fileContent = Files.readAllBytes(Paths.get(folderPathStr + "\\" + fileEntry.getName()));
                encodedString = Base64.getEncoder().encodeToString(fileContent);
                finalString = convertEncodeStringToBitArray2(encodedString);
                waktu = System.currentTimeMillis() - now;
                counter++;
            }
        }
        if (fileEntry.getParent().length() > 47) {
            DataContainer innerData = new DataContainer();
            innerData.setNmFolder(fileEntry.getParent());
            String namaFdrParent = fileEntry.getParent().substring((panjangChar + 1), fileEntry.getParent().length());
            innerData.setNmFolderParent(namaFdrParent);
            innerData.setNmFile(fileEntry.getName());
            innerData.setIsiFile(encodedString);
            innerData.setIsiByteFile(finalString);
            innerData.setWaktu(waktu);
            innerData.setPo("p");
            dic.inputData(innerData);
            finalString = null;
            waktu = 0;
        }
    }
}

public String convertEncodeStringToBitArray2(String encodedString) {
    StringBuilder innerFinalString = new StringBuilder();
    for(int i = 0; i < encodedString.length(); i++){
        int tempChar = (int)encodedString.charAt(i);
        innerFinalString.append(Integer.toString(tempChar,2));//& this is nc
    }
    return innerFinalString.toString();
}
```

Figure 3. Read file or folder and conversion code in Java

This work consists of several steps. At the first step we conduct some steps as describe below:

- Read fingerprints image file in every folder and we read one by one.
- Convert image data to array string type using Base64.
- Convert array string to binary code. This is a must because we use binary type in the swap or crossover process and mutation process.

- Population filtering to select parent candidates. In the first experiment, we use 20% of the population then we got swap or crossover result was too big. Caused by the limitation of our hard drive, then we decided to use only 2% of the population so that the swap results will not overwhelm the hard drive.
- Swap or crossover process, mating all of the 2% of the population to get the new offspring. The process is to mate every binary to each other then we got 4900 new offsprings.
- The last process is the mutation for 4900 new offsprings from the crossover process. This is a binary reverse process where each value of 1 will be reversed to 0 and 0 will be reversed to 1.

These codes above are to read every folder that contains fingerprints image data, then converted to array string type using the Base64 process and the result then converted to binary code because crossover and mutation process is mate and clone process of binary code. Figure 4 is the elitism code. Select randomly from 20% of the population to find parent candidates and because of the limitation of our hard drive we use only 2% of the population. The last *"null"* code is to empty the variable when it is not used so that RAM is not full.

Figure 5 is a crossover function code to mate every selected sample to get the new offsprings where the selected sample divided into 2 groups, male and female then swap to all samples to the other group and looped it. The swap process is depicted in Figure 6 starting from sample 1 in one group mated with all samples in the other group. Likewise with sample 2 and others. This process will be repeated continuously until all the samples have been successfully mated. The last process is the process of mutation. The program code is in Figure 7. Where the results of the marriage from the crossover process are cloned to then reverse the binary code, the value of 1 becomes 0 and vice versa, to get new offspring.

```
public void getElitsm(Integer randomSampling) {
    List<String> namaFolders = dic.getDataFolder();
    for (String nmFdr:namaFolders) {
        Integer hitungData = dic.getDataCountByteArrayFingerPrint(nmFdr);
        Double limitData = hitungData * (randomSampling.doubleValue() / 100);
        List<String> byteArrays = dic.getDataByteArrayFingerPrintRandom(nmFdr, limitData.intValue(), 0);
        for (String byteArr:byteArrays) {
            DataElitsmPre elitsPre = new DataElitsmPre();
            elitsPre.setNmFolder(nmFdr);
            String[] dataPre = byteArr.split("-");
            elitsPre.setKey(Integer.valueOf(dataPre[0]));
            elitsPre.setIsiByteFile(dataPre[1]);
            dic.inputDataPraElistm(elitsPre);
            dataPre = null;
            elitsPre = null;
        }
        byteArrays = null;
    }
}
```

Figure 4. Elitism code in Java

```
public void getCrossOver() {
    Integer hitungData = dic.getDataCountElitsmPre();
    Integer limit = hitungData / 2;
    List<DataElitsmPre> elitsmList = dic.getDataByteArrayAfterElitsm(limit, 0);
    List<DataElitsmPre> elitsmListInner = dic.getDataByteArrayAfterElitsm(limit, (limit * 1));
    for (DataElitsmPre elitsm:elitsmList) {
        int keyMale = elitsm.getKey();
        String male = elitsm.getIsiByteFile().substring(0, 50);
        for (DataElitsmPre elitsmInner:elitsmListInner) {
            long now = System.currentTimeMillis();
            StringBuilder strBuild = new StringBuilder();
            String married = strBuild.append(male).append(elitsmInner.getIsiByteFile()
                    .substring(50, elitsmInner.getIsiByteFile().length())).toString();
            int keyFemale = elitsmInner.getKey();

            DataContainer innerData = new DataContainer();
            innerData.setIsiByteFile(married);
            innerData.setPo("o");
            innerData.setParentOne(Integer.valueOf(keyMale).longValue());
            innerData.setParentTwo(Integer.valueOf(keyFemale).longValue());
            long waktu = System.currentTimeMillis() - now;
            innerData.setWaktu(waktu);
            dic.inputDataCrossOver(innerData);
        }
    }
    elitsmListInner = null;
    elitsmList = null;
}
```

Figure 5. Crossover code in Java language

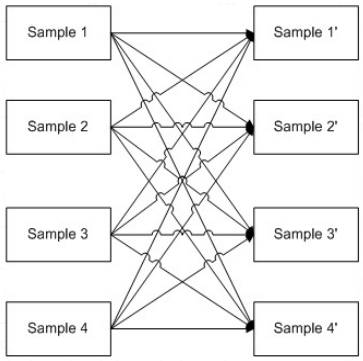


Figure 6. Crossover illustration

```
public void getMutationTwo() {
    List<Integer> keyList = dic.getKeyByteArrayCrossOver();
    for (Integer keys:keyList) {
        long now = System.currentTimeMillis();
        String mutationCand = dic.getByteArrayCrossOver(keys);
        String header = mutationCand.substring(0, 1000);
        String mutant = mutationCand.substring(1000, 1200);
        String footer = mutationCand.substring(1200, mutationCand.length());

        StringBuilder strReverse = new StringBuilder();
        char[] reverseMutant = mutant.toCharArray();
        for (int i = 0; i < reverseMutant.length; i++) {
            String name = (reverseMutant[i] == '0')?("1"):("0");strReverse.append(name);
        }
        StringBuilder strBuild = new StringBuilder();
        String married = strBuild.append(header).append(strReverse.toString()).append(footer).toString();

        DataContainer innerData = new DataContainer();
        innerData.setIsiByteFile(married);
        innerData.setPo("o");
        innerData.setParentOne(keys.longValue());
        long waktu = System.currentTimeMillis() - now;
        innerData.setWaktu(waktu);
        dic.inputDataMutation(innerData);
    }
}
```

Figure 7. Mutation code in Java

4. EXPERIMENTAL RESULT

According to Table 2, experiments carried out with 15 specimens of data. Once specimen consists of fingerprint data that grouped before. The following is a description of each specimen;
Specimen 1 consist of the entire fingerprint data
Specimen 2 consists of fingerprint data type A
Specimen 3 consists of fingerprint data type B
Specimen 4 consists of fingerprint data type C
Specimen 5 consists of fingerprint data type D
Specimen 6 consists of a combination of fingerprint data type A to type B
Specimen 7 consists of a combination of fingerprint data type A to type C
Specimen 8 consists of a combination of fingerprint data type A to type D
Specimen 9 consists of a combination of fingerprint data type B to type C
Specimen 10 consists of a combination of fingerprint data type B to type D
Specimen 11 consists of a combination of fingerprint data type C to type D
Specimen 12 consists of a combination of fingerprint data type A, B, and C
Specimen 13 consists of a combination of fingerprint data type A, B, and D
Specimen 14 consists of a combination of fingerprint data type B, C, and D
Specimen 15 consists of a combination of fingerprint data type A, C, and D

According to these specimens, we conduct four steps of experiment, consists of:

- Partial Test 1. In this part, we start with reading fingerprint image data, convert data to array string, and convert array string to binary code.
- Partial Test 2. This is a selection process which uses elitism principal. We select parent candidates 2% of the entire population.
- Partial Test 3. This part is a swap or crossover process to generate new offspring.
- Partial Test 4. This is the last where we do mutation to each offspring.

We measure every processing time in millisecond for each partial test then add them up. Following the results test specimen, 2 achieved the best result and has the most efficient processing time with 109168 ms. The specimen 1 is the worst compared to the entire specimen with total processing time 4032220 ms. This algorithm has processed every specimen successfully with an average of total processing time 1516986 ms.

Table 2. Numerical experiment result

		MA					Result Data				
		Partial Test (ms)				Total Time	Original	Elitism	Swap	Mutation	Data Storage
		1	2	3	4						
Specimen	1	710548	1972509	561290	787872	4032220	7200	140	4900	4900	22.8 GB
	2	68180	17167	6323	17498	109168	1800	35	289	289	270 MB
	3	241623	212153	43042	63946	560765	1800	35	289	289	4.8 GB
	4	256843	215656	45350	70968	588818	1800	35	289	289	4.3 GB
	5	171731	127978	27244	44162	371115	1800	35	289	289	2.4 GB
	6	301021	472539	207748	339992	1321301	3600	70	1225	1225	8.9 GB
	7	259104	363373	158062	279576	1060116	3600	70	1225	1225	7.9 GB
	8	211430	261118	115808	212052	800408	3600	70	1225	1225	4.5 GB
	9	469325	745542	159725	245119	1619711	3600	70	1225	1225	11.4 GB
	10	499744	646664	110142	176031	1432582	3600	70	1225	1225	8 GB
	11	390723	587601	109648	175632	1263607	3600	70	1225	1225	7.6 GB
	12	597502	1228740	416651	684167	2927060	5400	105	2704	2704	17.4 GB
	13	485254	1002529	307545	450989	2246318	5400	105	2704	2704	13.1 GB
	14	620749	1277471	244341	377135	2519696	5400	105	2704	2704	15.8 GB
	15	355975	805250	264398	476298	1901923	5400	105	2704	2704	12.3 GB

5. CONCLUSION

In this paper, we divide the image fingerprint data into 15 specimens to do four partial tests. In order to know the performance of MA, we measure every specimen in every partial test then generate each processing time. According to the results, specimen 2 achieves the best processing time with 1019168 ms, it has the most efficient processing time of all specimens. At the feature, we will conduct some experiments to this algorithm performance in many environmental systems, such as in a computer network.

ACKNOWLEDGEMENTS

This work is supported by Research and Technology Transfer Office, Bina Nusantara University as a part of Bina Nusantara University's International Research Grant entitled MEMETIC ALGORITHM IN HIGH-PERFORMANCE COMPUTATION with contract number: No.026/VR.RTT/IV/2020 and contract date: 6 April 2020.

REFERENCES

[1] A. K. Jain., *et al*., "Biometrics: personal identification in networked society," *Springer International*, 2006.
[2] D. Maltoni, *et al*., "Handbook of fingerprint recognition," *Springer-Verlag New York Inc,* 2009.
[3] Anil K. Jain, Jianjiang Feng, "Latent fingerprint matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, January 2011.
[4] S. Pankanti, Salil Prabhakar, Anil K. Jain, "On the individuality of fingerprints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 8, pp. 1010-1025.
[5] A. Jain, Lin Hong, "On-line fingerprint verification, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 3, September 1996.
[6] A. K. Jain, *et al*., "An identity-authentication system using fingerprints," *Proceedings of IEEE*, vol. 85, no. 9, pp. 1365-1388, September 1997.

[7] Pablo Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: toward memetic algorithms," *Technical Report, Caltech Concurrent Computation Program,* October 2000.
[8] Peter Mers, Bernd F., "Fitness landscapes and memetic algorithm design, *McGraw-Hill, London,* 1999.
[9] Yew-Soon Ong*, et al.,* "Classification of adaptive memetic algorithms: a comparative study," *IEEE Trans. Syst. Man. Cybern,* vol. 36, no. 1, pp. 141-152, February 2006.
[10] Andrea C., "A fast adaptive memetic algorithm for off-line and on-line control design of PMSM drivers," *IEEE Trans. Syst. Man Cybern. Part B*, vol. 37, no. 1, 00. 28-41, 2007.
[11] Licheng Jiau, *et.al*., "Natural and remote sensing image segmentation using memetic computing," *IEEE Comput. Intell. Mag,* vol. 5, no. 2, pp. 78-91, May 2010.
[12] Maren Urselman, *et.al*., "A memetic algorithm for global optimization in chemical process synthesis problems," *IEEE Transactions on Evolutionary Computation,* vol. 15, no.5, pp. 659-683, October 2011.
[13] Yanping Lu, *et.al.,* "Particle swarm optimizer for variable weighting in clustering high-dimensional data," *Machine Learning*, vol. 82, no. 1, pp. 43-70, May 2009.
[14] Liang Bai, *et.al*., "A novel attribute weighting algorithm for clustering high-dimensional categorical data," *Pattern Recognition*, vol. 44, no. 12, pp. 2843-2861,December 2011.
[15] Stefan Lang, *et.al*., "Fast extraction of neuron morphologies from large-scale SBFSEM image stacks," *J. Comput.* Neurosci, vol. 31, no. 3, pp. 533-545, March 2011.
[16] Yutong Zhao, Fu K. Sheong, "A fast paralel clustering algorithm for molecular simulation trajectories," *Journal of Computational Chemistry*, vol. 34, no. 2, pp. 95-104, January 2013.
[17] Silvia Bahmann, Jens Kortus, "EVO—Evolutionary algorithm for crystal structure prediction," *Comput. Phys. Commun*, vol. 184, no.6, pp. 1618-1625, June 2013.
[18] Kush R. Varshney, Alan S. Willsky, "Linear dimensionality reduction for margin-based classification: High-dimensional data and sensor networks," *IEEE Trans. Signal Process*, vol. 59, no.6, pp. 2496-2512, June 2011.
[19] Hakan Ergun, *et.al.,* Transmission system topology optimization for large-scale offshore wind integration," *IEEE Trans. Sust. Energy*, vol. 3, no. 4, pp. 908-915, October 2012.
[20] Jin-Hyuk Hong, Sung-Bae Cho, "Efficient huge-scale feature selection with speciated genetic algorithm," *Pattern Recogn. Lett*, vol. 27, no.2, pp. 143-150, January 2006.
[21] Dongxiao Niu, *et.al*., "Power load forecasting using support vector machine and ant colony optimization," *Expert Syst.* Appl, vol. 37, no. 3, pp. 2531-2539, March 2010.
[22] Yihan Shao, et.al., "Advances in methods and algorithms in a modern quantum chemistry program package," *Physical Chemistry Chemical Physics: PCCP*, vol. 8, no. 27, pp. 3172–3191, July 2006.
[23] Tapta K. Roy, Robert B. Gerber, "Vibrational self-consistent field calculations for spectroscopy of biological molecules: new algorithmic developments and applications," *Physical Chemistry Chemical* Physics, vol. 15, no. 24, pp. 9468-9492, May 2013.
[24] Jacques Blum, *et. al*., "Data assimilation for geophysical fluids," *P.G. Ciarlet (Ed.), Handbook of Numerical Analysis,* vol. 14, pp. 385–441, 2009.
[25] Joel T. Dudley, et.al., "Drug discovery in a multidimensional world: systems, patterns, and networks," *Journal of Cardiovascular Translational Research*, vol. 3, no. 5, pp. 438-447, October 2010.
[26] Weiliang Shi, Grace Wahba, Rafael A. Irizarry, "The partitioned LASSO-patternsearch algorithm with application to gene expression data," *BMC Bioinformatics*, vol. 13, no. 1, March 2012.
[27] Mitra Montazeri, "Memetic algorithm image enhancement for preserving mean brightness without losing image features," *International Journal of Image and Graphics*, vol. 19, no.4, October 2019.
[28] Manosij Ghosh, *et.al.,* "Feature selection for facial emotion recognition using late hill-climbing based memetic algorithm," *Multimed. Tools Appl*, June 2019.
[29] Rashmi Welekar, Nileshsingh V. Thakur, "An enhanced approach to memetic algorithm used for character recognition," *Third International Congress on Information and Communication Technology,* vol. 797, pp. 593-602, January 2019.
[30] B.W. Hwang, *et.al.*, "Feature selection for handwritten word recognition using memetic algorithm," *Advances in Intelligent Computing,* vol. 3644, November 2005.
[31] Manosij G., et.al., "Featue selection for handwriting word recognition using memetic algorithm," *Advances in Intelligent Computing*, vol. 3644, pp. 103-124, May 2018.
[32] B. Vinoth Kumar, *et.al*., "Evolutionary algorithm with memetic search capability for optic disc lokalization in retinal fundus images," *Challenges and Solutions Intelligent Data-Centric System*, pp. 191-207, 2019.
[33] Yipeng Zhang, Yanfei Zhong, "Sub-pixel mapping based on memetic algorithm for hyperspectral imagery," *IEEE International Geoscience and Remote Sensing Symposium,* July 2015.
[34] Mingyang Zhang, *et. al*., "Memetic algorithm based feature selection for hyperspectral images classification," *2017 IEEE Congr. Evol. Comput. CEC*, June 2017.
[35] Shaveta Dargan, Munish Kumar, "A Comprehensive survey on the biometric recognition systems based on physiological and behavioral modalities," *Expert Systems With Applications*, vol. 143, April 2020.
[36] W. Sheng, G. Howells, M. Fairhurst, and F. Deravi, "A memetic fingerprint matching algorithm," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 402-411, 2007.
[37] Lee H.C. & Gaensslen R.E, "Advances in Fingerprint Technology," *2nd edition, Elsevier, New York*, 2001.
[38] Andre A., "Fingerprint Techniques," *Chilton Book Company, London,* 1971.

BIOGRAPHIES OF AUTHORS

Priati Assiroj, was born in Cirebon, Jawa Barat, Indonesia in 1986. She has Bachelor and Master in Computer Science. She received the Bachelor from STMIK Bani Saleh Bekasi, in 2011and received her Master from STMIK LIKMI, Bandung, Indonesia, in 2016. From 2014 to 2016, she was a lecturer in Universitas Singaperbangsa Karawang, Indonesia and from 2016 and recent she is a lecturer in Universitas Buana Perjuangan Karawang in Information System Dept. Since January 2019 she is a lecturer in Politeknik Imigrasi, Ministry of Law and Human Right, Republic of Indonesia. Since March 2018 she has been a scholar of Bina Nusantara Graduate Program, Doctor of Computer Science, Bina Nunsantara University Jakarta, Indonesia. Her research fields are data mining, high performance computing and evolutionary algorithm.

H. L. H. S. Warnars, received a Ph.D. degree in Computer Science from Manchester Metropolitan University. Since September 2015 he is a Head of Information Systems concentration at department Doctor of Computer Science Bina Nusantara University, works some project research with my doctoral computer Science students in research area such as Game, Artificial Intelligence including Data Mining, Machine Learning and Decision Support System application such as DSS, BI, Dashboard, Data Warehouse, and so on

Edi Abdurrachman, received B. Sc and Master of Statistics in Applied Statistics from Bogor Agricultural University, then received M.Sc and Ph.D. in survey statistics and statistics from IOWA State University, USA. He is currently a professor and dean of the Binus Graduate Program, Doctor of Computer Science, Bina Nusantara University Jakarta. His research interest includes statistics, survey statistics, and applied statistics and management information systems. Mr. Abdurrachman's awards and honors include the MU SIGMA RHO Society (1985) and Best Lecturer Binus University (2012). He is also a member of the American Statistical Association, International Association of Engineers (IAENG), Gamma Sigma Beta, and as a Vice President of the Asian Federation for Information Technology in Agriculture. From 1980-2015 actives in the ministry of agriculture in many positions of director. He is also active as a public speaker in national and international seminars.

Achmad I. Kistijantoro, received the B.Eng. degree in informatics from the Institute of Technology Bandung, (ITB), Bandung, Indonesia, the masters' degree from TU Delft, Delft, The Netherlands, and the Ph.D. degree from the University of Newcastle upon Tyne, Newcastle upon Tyne, U.K., His current research interests includes distributed systems, parallel computation, and high-performance computation.

Antoine Doucet is a Full Professor in computer science at the L3i laboratory of the University of La Rochelle since 2014. He leads the research group in document analysis, digital contents and images (about 40 people) and is additionally the director of the ICT department of the Vietnam-France University of Science and Technology of Hanoi. Additionally, he is the principal investigator of the H2020 project NewsEye, running until 2021 and focusing on augmenting access to historical newspapers, across domains and languages. He further leads the effort on semantic enrichment for low-resourced languages in the context of the H2020 project Embeddia. His main research interests lie in the fields of information retrieval, natural language processing and (text) data mining. The central focus of his work is on the development of methods that scale to very large document collections and that do not require prior knowledge of the data, hence that are robust to noise (e.g stemming from OCR) and language-independent. Antoine Doucet holds a PhD in computer science from the University in Helsinki (Finland) since 2005, and a French research supervision habilitation (HDR) since 2012.