A publication of ISCA*:
International Society for Computers
and Their Applications

INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS

TABLE OF CONTENTS

*** "International Journal of Computers and Their Applications is abstracted and indexed in INSPEC and Scopus."**

Volume 22, No. 1, March 2015 **ISSN 1076-5204**

International Journal of Computers and Their Applications

A publication of the International Society for Computers and Their Applications

ISCA Headquarters···•···64 White Oak Court, Winona, MN 55987···•···Phone: (507) 458-4517
E-mail: isca@ipass.net • URL: http://www.isca-hq.org

Editor's Note: March 2015

It is my distinct honor, pleasure and privilege to serve as the Editor-in-Chief of the International Journal of Computers and Their Applications (IJCA). I have a special passion for the International Society for Computers and their Applications.

I would like to begin this volume by giving a review of this past year. In 2014 we had 67 articles submitted to the International Journal of Computers and Their Applications. We currently have 15 that are still under review. As a reminder, authors are now charged $50 per page and the first two pages are free for ISCA members, and the journal will not be accepting articles that are less than 6 pages. The authors of these papers will be encouraged to submit their papers to ISCA conferences.

I look forward to working with everyone in the coming years to maintain and further improve the quality of the journal. I would like to invite you to submit your quality work to the journal for consideration of publication. I also welcome proposals for special issues of the journal. If you have any suggestions to improve the journal, please feel free to contact me.

Frederick C. Harris, Jr.
Computer Science and Engineering
University of Nevada, Reno
Reno, NV 89557, USA
Phone: 775-784-6571
Email: Fred.Harris@cse.unr.edu

This year we have 4 issues planned (March, June, September, and December). We begin with a special issue from the best papers at the ISCA Fall Conference cluster (CAINE, and SEDE). We have a proposal for the best papers from the ISCA Spring Conference cluster (CATA/BICOB) which will appear in the September issue. The other two issues (June and December) are being filled with submitted papers.

I would also like to announce that I begun a search for a few Associate Editors to add to our team. There are a few areas that we would like to strengthen our board with, such as Image Processing. If you would like to be considered, please contact me via email with a cover letter and a copy of your CV.

Frederick C Harris, Jr.
Editor-in-Chief
Email: Fred.Harris@cse.unr.edu

Guest Editorial:
Special Issue from ISCA Fall-2014 Conferences

This Special Issue of IJCA is a collection of five refereed papers selected from the following ISCA conferences (co-located at the Holiday Inn Downtown Superdome, New Orleans, Louisiana, USA, October 13{15, 2014):

- CAINE 2014: 27th International Conference on Computer Applications in Industry and Engineering
- SEDE 2014: 23rd International Conference on Software Engineering and Data Engineering

Each paper submitted to the conferences was reviewed by at least two members of the International Program Committee, as well as by additional reviewers, judging the originality, technical contribution, significance and quality of presentation. After the conferences, nine best papers were recommended by the Program Committee members to be considered for publication in this Special Issue of IJCA. The authors were invited to submit a revised version of their papers. After extensive revisions and a second round of review, five papers were accepted for publication in this issue of the journal.

The papers in this special issue cover a wide range of research interests in the community of computers and applications. The topics and main contributions of the papers are briefly summarized below.

Zulfiqar Ali, Ghulam Muhammad, Mansour Alsulaiman, and Khalid Al-Mutib of King Saud University, Saudi Arabia, and Irraivan Elamvazuthi of Universiti Teknologi Petronas, Malaysia, introduced a speech recognition method called oriented local features (OLF) for feature extraction. They used Hidden Markov model for modeling in the training phase. The proposed approach was applied to a speech recognition system for dysphonic patients. Their experiments show that the proposed method achieved about 95% recognition accuracy.

Bidyut Gupta, Sindoora Koneru, and Shahram Rahimi of Southern Illinois University at Carbondale, USA, presented novel approaches (pseudo sub-diameter for static core selection and super pseudo sub-diameter for group-based core selection) for networks that use LSR as the unicast routing protocol. The presented methods select more than one core to achieve fault tolerance. Simulation results have shown the superiority of the proposed approach over some other approaches.

Antoine Bossard of Tokyo Metropolitan University, Japan, and Keiichi Kaneko of Tokyo University of Agriculture and Technology, Japan, presented an algorithm to solve the node-to-set disjoint paths routing problem in a Torus-Connected Cycles (TCC) network topology as the disjoint paths routing problem is critical for the reliability and performance of parallel systems. The authors provided theoretical analysis of the complexity of the algorithm.

Hamwira Yaacob, Wahab Abdul of International Islamic University Malaysia, and Norhaslinda Kamaruddin of MARA University of Technology, Malaysia, proposed a feature extraction technique using both electroencephalogram (EEG) signals and the behavior of neural activations for emotion classification. The technique is based on the Cerebellar Model Articulation Controller (CMAC) model. The emotion classification process uses Evolving Fuzzy Neural Network (EFuNN).

Yong Shi of Kennesaw State University, USA, studied the dynamic characteristics (distribution) of multidimensional data and presented an algorithm to dynamically change the set of clusters and the set of outliers as the data set changes. His experiments on synthetic data sets showed an improved accuracy in comparison to some existing approaches such as CURE.

As guest editors we would like to express our genuine appreciation for the encouragement and support from the editor-in-chief of IJCA, Frederick C. Harris, Jr. We also owe many thanks to the authors and program committees of the conferences from which these papers were selected.

We hope you enjoy this special issue of the IJCA and we look forward to seeing you at a future ISCA conference. More information about the ISCA society can be found at http://www.isca-hq.org.

Guest Editors:

Gongzhu Hu, Central Michigan University, USA, CAINE 2014 Conference Chair
Yan Shi, University of Wisconsin-Platteville, USA, SEDE 2014 Program Chair

January 2015

Oriented and Interpolated Local Features for Speech Recognition of Vocal Fold Disordered Patients

Zulfiqar Ali[*,†], Ghulam Muhammad[*], and Mansour Alsulaiman[*]
King Saud University, Riyadh, SAUDI ARABIA

Irraivan Elamvazuthi[†]
Universiti Teknologi PETRONAS, Perak, MALAYSIA

Khalid Al-Mutib[**]
King Saud University, Riyadh, SAUDI ARABIA

Abstract

A novel technique of oriented local features (OLF) for speech recognition has been introduced in this paper. A speech recognition system for dysphonic patients is implemented by application of the proposed technique. The developed system is evaluated by performing the training in three different ways: (a) with pathological samples, (b) with normal samples, and (c) with pathological and normal samples together. We compare the performance of the proposed feature with the most widely used speech feature in speech recognition, i.e., Mel-frequency cepstral coefficients. The Hidden Markov model is used for recognizing the speech. The proposed technique achieved a 94.98% recognition rate, which is almost identical to the recognition rate of 95.45% obtained with MFCC.

Keywords: Vocal fold disorders, automatic speech recognition, Arabic digits, MFCC, HMM.

1 Introduction

The number of dysphonic patients having different types of voice disorders has increased significantly. In the United States, approximately 7.5 million people have vocal difficulty [14]. It has been found that 15% of the total visitors to the King Abdul-Aziz University Hospital complain from a voice disorder [16]. The complications caused by a voice problem in a teaching professional are significantly greater than in a non-teaching professional. Studies revealed that, in the U.S., the prevalence of voice disorders during a lifetime is 57.7% for teachers and 28.8% for non-teachers [17]. Approximately, 33% of teachers in the Riyadh area suffer from voice disorders [10].

At Communication and Swallowing Disorders Unit, King Abdul Aziz University Hospital, a high volume of voice disorder cases is examined (almost 760 cases per annum) in individuals with various professional and etiological backgrounds.

An automatic speech recognition system for assessment or therapy of a voice disorder is suggested in [8]. The recognition rate of a speech recognition system can be affected by size of the vocabulary, continuity of speech (isolated words or continuous speech), the microphone quality and its placement, and text-dependence. Speech varies widely from person to person, even if pronunciation is correct. For example, a small change in pitch could result in a rejection by a computer. Extensive work has been done on speech recognition systems (SRS) for normal speakers [12, 18-19] but few studies have examined speech recognition in dysphonic patients [2, 13]. In [13] the patients have six voice disorders: Cyst, LPRD, SD, Sulcus, Nodules or Polyp. Mel-frequency cepstral coefficients (MFCC) features alone and with delta coefficients are extracted from the Arabic digits and inputted to GMM. The system is trained by normal speakers only, but tested with both normal and pathological samples. The recognition rate for normal subjects is 100%, and varies from 56% to 82.50% for disorder samples. Another automatic SRS is presented in [2]. The database contains 50 male and 21 female patients. MFCC, Linear predictive cepstral coefficients (LPCC), PLP (Perceptual Linear Prediction) [4] and RASTA_PLP (Relative Spectral Transform Perceptual Linear Predictive) [5-6] are extracted from the speech samples of the dysphonic patients and inputted to the pattern matching techniques of Hidden Markov model (HMM) and Gaussian mixture model (GMM).The techniques of HMM and GMM are implemented with various numbers of states and Gaussian's mixtures. The database is divided into three parts: male only, female only and mixed gender. The speech recognition systems developed in the studies [13] and [2] is based on the word model, where whole word is used for the training of the system. In [9], Multi-directional local features are used for automatic speaker recognition system.

In this paper, an automatic phoneme based speech recognition system is implemented by using MFCC and the

[*] Department of Computer Engineering, College of Computer and Information Sciences. E-mail: {zuali, ghulam, msuliman}@ksu.edu.sa.
[†] Centre for Intelligent Signal & Imaging Research, Department of Electrical and Electronic Engineering. E-mail: zulfiqar_g02579 @utp.edu.my, irraivan_elamvazuthi@petronas.com.my.
[‡] Department of Software Engineering. Email: muteb@ksu.edu.sa.

proposed Oriented Local Features (OLF). The HMM [1, 3, 11] is implemented with the means of Hidden Markov model Toolkit (HTK) [7]. The proposed OLF provided good recognition rate, and obtained results comparable with conventional features MFCC. In the developed system, a word is divided into its phonemes. The HMM is built using a phoneme based model.

The rest of the paper is organized as follows: Section 2 describes the developed system, including different speech features. Section 3 focuses on experimental results and provides analysis on the results. Finally, Section 4 highlights some conclusions.

2 Phoneme Based Automatic Speech Recognition System

A speech recognition system has two major phases: first is the training phase and the second is testing of the developed system. The output of the first step is acoustic models of each word in the vocabulary, while the second step uses the generated models and makes a comparison of a test utterance with each of the models. The model of a word having maximum likelihood with the test utterance is the recognized word.

The first phase consists of two important components: 1) feature extraction and 2) acoustic model generation. The features we used are MFCC plus energy, oriented local features (OLF), and interpolated OLF. The modeling technique is Hidden Markov model.

In the phoneme based system, each word of the vocabulary is split into its phonemes. Arabic digits with their IPA symbols are listed in Table 1. This kind of system generates accurate acoustic models; hence, produces good recognition rates. The following subsections will describe the pre-emphasis of the signal, and the feature extraction techniques.

Table 1: Arabic Digits with their IPA Symbols

Digits		IPA
Symbol	Arabic	
1	واحد	/w/, /a/, / ħ/, /i/, /d/
2	إثنان	/i/, /th/, /n/, /a/, /y/, /n/
3	ثلاثة	/th/, /a/, /l/, /ā/, /th/, /a/
4	أربعة	/a/, /r/, /b/, /?/, /a/
5	خمسة	/kh/, /a/, /m/, /s/, /a/
6	ستة	/s/, /i/, /t/, /t/, /a/
7	سبعة	/s/, /a/, /b/, /?/, /a/
8	ثمانية	/th/, /a/, /m/, /ā/, /n/, /y/, /a/
9	تسعة	/t/, /i/, /s/, /?/, /a/

2.1 Mel-Frequency Cepstral Coefficients

Before extracting the features, the speech samples are passed through a pre-emphasis filter to increase the magnitude of high frequencies with respect to the magnitude of lower frequencies. The pre-emphasis filter is implemented by using

$$y(n) = x(n) - \alpha \ . \ x(n-1) \tag{1}$$

Where $x(n)$ and $y(n)$ are the input and output signals, respectively, and $\alpha = 0.95$.

MFCC simulates human auditory mechanism and performs reasonably well under robust conditions. Figure 1 shows a block diagram of the MFCC calculation. First, digitized wave data is divided into overlapping frames, where the frame length is 20 milliseconds. This division is needed to analyze the speech in small pseudo-stationary segments.

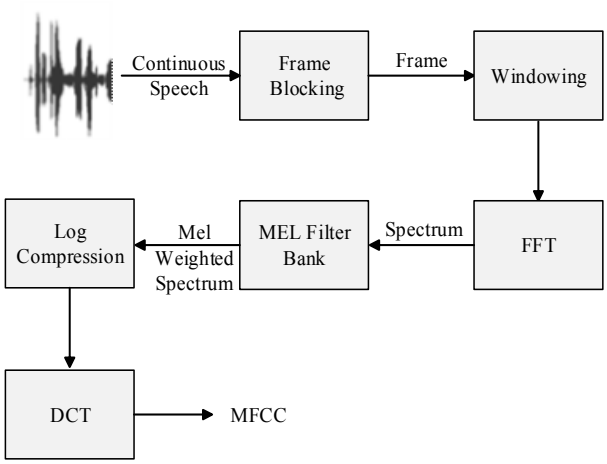


Figure 1: Block diagram for MFCC extraction

The resultant frame is multiplied by an N-point Hamming window to minimize the effect of spectral leakage. The Hamming window has almost zero values towards the both ends ensuring the continuity of the signal in successive frames.

The windowed signal, s_{nw}, is obtained as

$$s_{nw} = \left\{ 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1} \right) \right\} s_n' \tag{2}$$

$$where \quad 0 \le n \le N-1$$

Fourier transformation (FT) is applied to the windowed signal to convert the time-domain signal into a frequency-domain signal (spectrum). Triangular band-pass filters (BPFs) are applied to divide the spectrum into certain frequency bands. The center frequencies of the BPFs are spaced on a Mel-scale, and the bandwidths correspond to well-known auditory perception phenomena called critical bandwidth. The relation between Mel-scale and linear scale (Hz), given by Equation 3, is almost linear up to around 800 Hz, and logarithmic beyond that

$$m = 2595 \log_{10}\left(1 + \frac{f}{700} \right), \quad m = 1,2,...,P \tag{3}$$

In Equation 3, m corresponds to Mel's index (P filters are used) and f refers to frequency in Hz. Therefore, by applying 26 BPFs (P = 26), N points of the spectrum are converted to only 26 values. Log is applied to the 26 outputs to make the convolution components additive and to adjust the dynamic

range in the spectrum. In this way, source excitation signal and vocal tract filter response become additive. The log outputs are then passed through discrete cosine transform (DCT) to de-correlate the components and reduce the dimension. The output of DCT is called MFCC. Typically, the 0-th coefficient is ignored and the first to 12 coefficients are retained to represent 12 MFCC.

First and second orders derivatives are calculated on these 12 components to find velocity and acceleration coefficients. These derivatives are normally extracted using linear regression as

$$\Delta_t = \frac{\sum_{i=1}^{B} i(c_{t-i,m} - c_{t+i,m})}{2\sum_{i=1}^{B} i^2} \quad (4)$$

Where, Δt corresponds to a velocity component at t-th frame, $C_{t,m}$ stands for m-th MFCC at t-th frame, and B is the length of the regression window.

In our work, we add the energy to the MFCC coefficients. The energy of a signal, $s(n)$, for N number of samples is defined as

$$E_s = \sum_{n=1}^{N} |s(n)|^2 \quad (5)$$

2.2 Oriented Local Features

Oriented local features (OLF) are calculated by applying 3-point linear regression along directions 0°, 22.5°, 45°, 67.5°, and 90° in the time frequency plane. In calculating OLF, pre-processing, frame blocking, windowing, Fourier's transformation, Mel filter bank, and log compression are the same as described in the section of MFCC. A 3-point linear regression (LR) is then applied on the log of Mel weighted spectrum (LMS) to obtain local features in a specific direction. In each direction, OLF is calculated separately. The application of LR on the 7x7 window of LMS matrix to calculate OLF in different directions is depicted in Figure 2(a)-(e). After applying LR, 12 coefficients from each frame are considered. In total, for all five directions, we will have 60 features. In other words, if a voice sample has 100 frames, then each frame will have 12 features in each direction, and dimensions of OLF will become 100x60.

Three-point LR is calculated by using following formula

$$LR = \frac{\sum_{r=1}^{3} i\left(L_{c+r} - L_{c-r}\right)}{2\sum_{r=1}^{3} r^2} \quad (6)$$

L_c is the center element of the selected 7x7 window of LMS matrix. L_{c+r} and L_{c-r} represent next and previous elements of

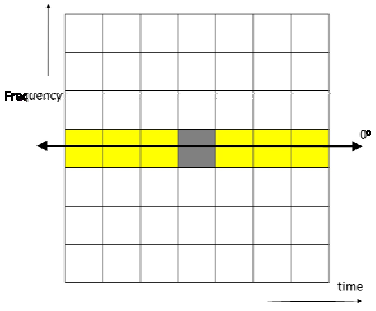


Figure 2(a): OLF at 0°

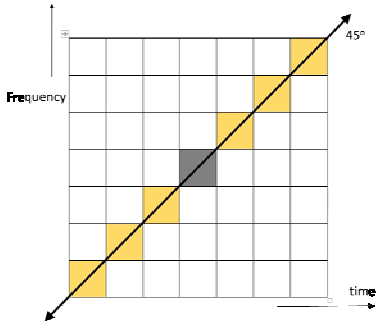


Figure 2(b): OLF at 45°

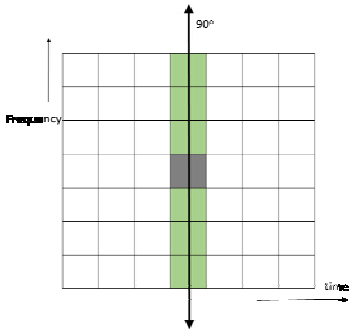


Figure 2(c): OLF at 90°

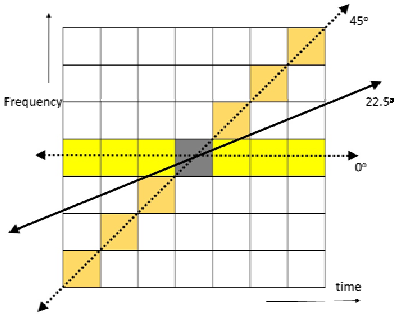


Figure 2(d): OLF at 22.50°

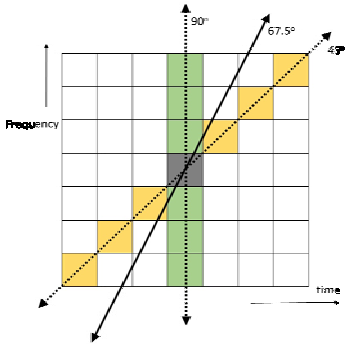


Figure 2(e): OLF at 67.50°

Figure 2(a)-(e): OLF at different angles

L_c in the direction where *LR* is going to be calculated. For each value of the LMS, *LR* is calculated, and it replaces the value. The *LR* for the values at boundaries is calculated by padding the LMS matrix with boundary columns and rows.

To get OLF at 22.5° and 67.5°, the points for *LR* at 22.5°, 67.5°, 202.5° and 247.5° are interpolated by using the points of 0°, 45°, 90°, 135°, 180°, 225°, 270°, 315° and 360°. The interpolation of the points along radius 1 is depicted in Figure 3. Similarly, interpolations along radius 2 and 3 are calculated. The interpolated data is obtained by using the formula of cubic spline interpolation [15].

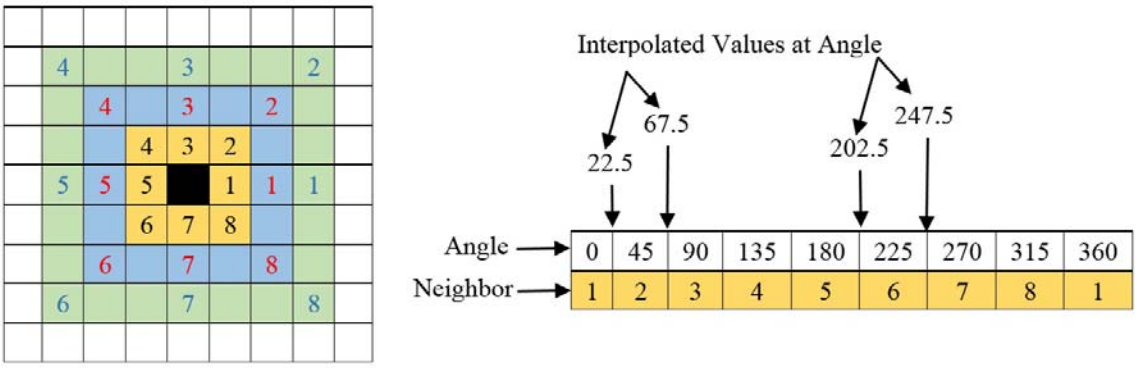


Figure 3: Interpolation of the values along radius 1

LR at 0° and 90° correspond to change in time and frequency, respectively. LR at 45°, correspond to change in both time and frequency. Similarly, *LR* along 22.5° and 67.5° also represent the change in time and frequency.

3 Experimental Results and Analysis

To evaluate the performance of the phoneme based speech recognition system, the developed system is evaluated with a database containing 142 subjects, 71 dysphonic patients and 71 normal persons. The dysphonic patients have five types of voice disorder. The distribution of samples among various disorders is provided in Table 2.

Each subject has uttered Arabic digits from one to nine. Therefore, the total number of samples in the database is 1278. Fifty percent of the normal and pathological samples are used for training the system and the remaining 50% samples are used for testing.

Table 2: No. of speakers for both types of subjects

Types of subject	Types of disorder	No. of subjects	Total
Dysphonic	Cyst	12	71
	GERD	22	
	Paralysis	6	
	Polyp	10	
	Sulcus	21	
Normal	--	71	71
Total			**142**

The experiments are performed by using 2, 4, 8, 12, 16, 20, 32, 50 and 64 Gaussian mixtures in each state, where numbers of states in HMM are three. The size of database is not very large. Therefore, we do not use a high number of states because data may not converge.

The system is evaluated in three different ways: (1) training and testing is performed with pathological samples (P - P), (2) training with normal and testing with pathological samples (N - P), and (3) training and testing is done with both types of the samples (NP - NP).

3.1 Speech Recognition with MFCC and Energy

In this subsection, speech recognition results are obtained by using the combination of energy and MFCC features. The experiments are performed with 13 features (static coefficients), 26 features (13 static and 13 delta coefficients), and 39 features (13 static, 13 delta, and 13 delta-delta coefficients). The bold values in each table represent the maximum recognition rate.

3.1.1 Recognition Rate with P – P. Firstly, the developed system is evaluated by using pathological samples for training and testing. The recognition rates (RR) for the different number of features are provided in Table 3. The maximum achieved RR is 91.22% with 39 features.

3.1.2 Recognition Rate with N – P. Secondly, the system is trained with normal samples and tested with disordered samples. The results show that performance of the system is improved by doing the training with normal samples and RR improved to 94.51%. The results are listed in Table 4.

3.1.3 Recognition Rate with NP – NP. Lastly, the system performance is evaluated by considering both types of samples, i.e., normal and pathological, for the training. The results are further improved and are presented in Table 5. The highest

Table 3: Recognition rates with P - P

No. of GMM	Digit recognition for P - P with different no. of features		
	13	*26*	*39*
2	**88.4**	**90.91**	90.91
4	87.77	90.6	**91.22**
8	86.83	90.28	89.66
12	86.52	89.66	90.28
46	88.09	89.34	89.34
20	85.89	87.77	88.09
32	86.21	89.97	87.77
50	86.52	84.95	85.89
64	86.83	86.52	81.19

Table 4: Recognition rates N - P

No. of GMM	Digit recognition for N - P with different no. of features		
	13	*26*	*39*
2	87.93	93.73	93.26
4	87.15	94.04	93.89
8	85.58	94.36	93.89
12	87.3	94.04	93.57
46	**88.24**	93.89	93.89
20	87.15	93.42	**94.51**
32	87.62	92.95	93.26
50	85.89	93.89	93.1
64	86.52	**94.51**	34.64

Table 5: Recognition rates with NP - NP

No. of GMM	Digit recognition for NP – NP with different no. of features		
	13	**26**	**39**
2	91.07	94.83	94.2
4	92.48	95.14	94.51
8	92.95	94.51	94.98
12	91.85	95.14	**95.45**
46	92.63	94.51	94.83
20	92.32	94.51	53.76
32	92.79	94.51	95.14
50	91.22	93.89	94.83
64	91.69	**95.45**	94.98

obtained recognition rate is 95.45% and is achieved with 26 features by using 64 Gaussians, and 39 features by using 12 Gaussians.

The confusion matrix for 39 features and 12 Gaussian, which provided the highest recognition rate for Energy + MFCC, is provided in Table 6. Diagonal elements represent the number of truly recognized samples, while all other values correspond to the number of misclassified samples. In the last column, the recognition rate for each digit is provided. The highest obtained recognition rate is for digits 5 and 7, and it is 98.59%. The trend of the recognition rate of all digits is shown in Figure 4.

To see the effect of the number of features, few experiments are performed with 20 features. The first and second derivatives are also calculated and appended, making the total number of features 40 and 60, respectively. The RR is listed in Table 7.

3.2 Speech Recognition with Oriented Local Features

Speech recognition is also performed with OLF features. In each direction, we have 12 coefficients. The experiments are done by combining the features of different directions. The experiments are performed by training the system with normal samples, and then by using both types of the samples. The case of training with pathology samples is not considered, as it did not provide good results in case of MFCC.

3.2.1 Recognition Rate at 0+45 with N – P and NP – NP. Table 8 provides the recognition rate for the combination of 0 and 45 degree (12 features in each direction). The highest result, when trained with normal samples is 92.01%, while a RR of 94.98% is achieved when the system is trained with both type of samples.

3.2.2 Recognition Rate with NP – NP. The experiments are performed by combining the features of different directions. The system is trained with both type of samples as it provided good results for the combination 0+45.

The highest RR obtained for the combination of 0 and 90 degree (0+90) is 94.51%, for 0+45+90 is 92.63%, and for 0+45+90+135 is 92.95%. The results are provided in Table 9.

Table 6: Confusion matrix of energy+MFCC for 39 features and 12 Guassians

Digits	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**RR(%)**
1	67	1	2	0	0	0	0	0	0	95.71
2	0	66	4	0	0	0	1	0	0	92.96
3	0	4	65	1	0	0	1	0	0	91.55
4	0	1	1	67	1	0	0	0	1	94.37
5	0	0	0	0	70	1	0	0	0	98.59
6	0	0	0	0	0	69	2	0	0	97.18
7	0	0	0	0	0	0	70	1	0	98.59
8	0	3	1	0	0	0	0	66	1	92.96
9	0	0	0	0	1	0	1	0	69	97.18

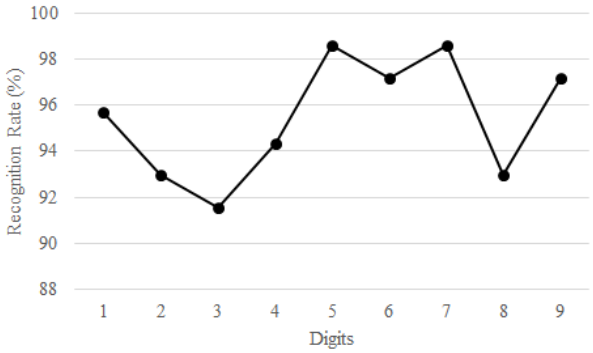


Figure 4: The trend of recognition rate of all digits for 39 features and 12 Gaussians

Table 7: Recognition rates with NP - NP with 20, 40 and 60 features

No. of GMM	Digit recognition for NP – NP with different no. of features		
	20	**40**	**60**
2	89.18	**94.04**	**93.89**
4	**90.75**	93.73	**93.89**
8	90.13	94.36	93.26
12	90.28	92.79	93.42
46	90.6	92.63	92.79
20	88.87	92.32	93.1
32	90.13	93.57	92.63
50	90.13	89.81	92.16
64	89.81	92.48	91.69

Table 8: Recognition rates with OLF at 0+45

No. of GMM	Digit recognition at *0+45*	
	N - P	**NP-NP**
2	91.69	92.79
4	91.22	93.1
8	91.07	94.67
12	91.07	94.83
16	**92.01**	94.83
20	91.54	93.73
32	90.91	93.89
50	88.71	94.04
64	90.6	**94.98**

Table 9: Recognition rates for NP - NP with OLF

No. of GMM	Digit recognition with NP - NP at different orientations		
	0 + 90	**0 + 45 +90**	**0+45+90+135**
2	94.04	91.54	91.54
4	94.2	92.01	92.79
8	**94.51**	92.32	**92.95**
12	93.1	91.38	91.22
46	92.79	92.16	91.22
20	93.42	**92.63**	91.54
32	92.95	91.38	89.81
50	90.75	89.81	89.81
64	93.57	89.66	88.56

3.3.2 Best Performance of OLF. The highest recognition rate with OLF is 94.98% obtained with a combination 0+45 by using 64 Gaussian mixtures, when the system is trained with both type of samples.

To show the number of truly recognized samples, a confusion matrix is provided in Table 10. All diagonal elements represent the number of truly classified samples for each digit. The elements other than diagonal are number of misclassified samples. The last column of Table 10 contains individual recognition rates for each digit; they are also plotted in Figure 5.

3.3 Speech Recognition with Interpolated OLF

OLF are interpolated in two directions, 22.50 and 67.50 degree. These features are used in two combinations *0+22.5+45* and *0+22.5+45+67.5*, when training is done by normal samples. The results are presented in Table 11. The highest obtained RR in this case is 89.03%.

Interpolated OLF are also used in *0+22.5+45* and *45+67.5+90*, when training is done by both types of samples. The RR for these combinations are listed in Table 12. The maximum achieved result is 93.57% for this case.

The overall highest obtained rate for interpolated OLF is 93.57% with *0+22.5+45* by using four Gaussian. The recognition rate for each digit is depicted in Figure 6.

Table 10: Confusion matrix of OLF for 0+45 with 64 mixtures

Digits	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**RR(%)**
1	66	2	1	1	0	0	0	0	0	94.29
2	0	68	1	0	0	1	1	0	0	95.77
3	0	2	66	2	0	0	0	1	0	92.96
4	0	1	1	65	1	0	3	0	0	91.55
5	0	0	0	0	69	1	0	0	1	97.18
6	0	0	0	0	0	69	2	0	0	97.18
7	0	1	0	1	0	0	68	1	0	95.77
8	0	3	0	0	0	0	0	67	1	94.37
9	0	1	0	0	0	1	1	0	68	95.77

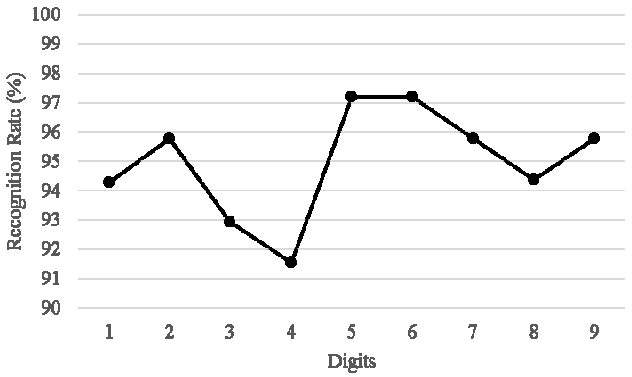


Figure 5: The trend of recognition rate of all digits for 0+45 with 12 Gaussians

Table 11: Recognition rates for N - P with interpolated OLF

No. of GMM	Digit recognition at different orientations	
	0 + 22.5 + 45	0+22.5+45 + 67.5
2	**89.03**	84.33
4	89.03	87.46
8	88.09	**87.77**
12	88.87	87.62
16	87.93	84.8
20	87.15	84.48
32	87.46	84.64
50	86.68	84.64
64	85.27	82.45

Table 12: Recognition rates for NP - NP with interpolated OLF

No. of GMM	Digit recognition at different orientations	
	0 + 22.5 + 45	45 + 67.5+90
2	91.38	85.97
4	**93.57**	91.15
8	91.54	89.58
12	92.32	**91.46**
16	92.01	88.48
20	93.10	89.74
32	91.85	89.11
50	89.03	85.5
64	90.44	88.33

4 Conclusion

A new type of speech features, OLF, are proposed in the study. A speech recognition system for the dysphonic patients is developed by application of the proposed features. The performance of the features is evaluated by carrying out different experiments on a database containing both pathological and normal samples. The maximum accuracy is provided by the combination of energy and MFCC, and it is 95.45%. The maximum accuracies achieved by OLF and interpolated OLF are 94.98% and 93.57%, respectively.

The obtained accuracy of 95% with proposed features is encouraging and comparable with the existing features. The performance of the developed system can be improved by modifying the proposed features OLF, so that the system could be applied in real application to benefit the dysphonic patients. By doing so they will be able to take the advantages like normal persons by using speech recognition applications.

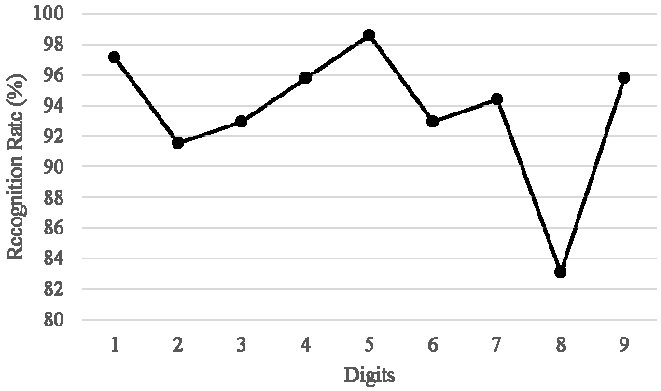


Figure 6: The trend of recognition rate of all digits for 0+22.5+45 with 4 Gaussians

Acknowledgment

This research was supported by NSTIP strategic technologies program number 12-MED2474-02 in the Kingdom of Saudi Arabia. The authors are thankful for this support.

References

[1] Z. Ali, M. Aslam., M. E. Ana María, and E. Gonzalo, "Text-Independent Speaker Identification using VQ-HMM Model Based Multiple Classifier System," *Lecture Notes in Computer Science*, Springer, 6438:116-125, 2010.

[2] M. Alsulaiman, "Speech Recognition for Medically Disordered Voice," *Proceedings of 24th International Conference on Computers and Their Applications in Industry and Engineering*, CAINE, pp. 233-237, 2011.

[3] L. E. Baum and T. Petrie, "Statistical Inference for Probabilistic Functions of Finite State Markov Chains," *Ann. Math. Stat.*, 37:1554-1563, 1966.

[4] H. Hermansky, "Perceptual Linear Predictive (PLP) Analysis for Speech," *J. Acoust. Soc. Am.*, 87(4):1738-1752, 1990.

[5] H. Hermansky, "Auditory Model for Parameterization of Speech in Real-Life Environment Based on Re-integration of Temporal Derivative of Auditory Spectrum," US WEST Advanced Technologies Research Report, File Folder ST 04-01, October 1990.

[6] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, "Compensation for the Effect of the Communication Channel in Auditory-Like Analysis of Speech (RASTA-PLP)," *Proc. of Eurospeech '91*, pp. 1367-1371, Genova, Italy, 1991.

[7] *Hidden Markov Model Toolkit (HTK)*, Version 3.4.1, Machine Intelligence Laboratory, Cambridge University Engineering Department, UK., Available at http://htk.eng.cam.ac.uk/download.shtml, 2009.

[8] P. Kitzing, A. Maier, and V. L. A Hlander, "Automatic Speech Recognition (ASR) and its use as a Tool for Assessment or Therapy of Voice, Speech, and Language Disorders," *Logopedics Phoniatrics Vocology*, 34:91-96, 2009.

[9] Awais Mahmood, Mansour Alsulaiman, Ghulam Muhammad, "Automatic Speaker Recognition Using Multi-Directional Local Features (MDLF)", *Arab J Sci Eng*, 39:3799-3811, 2014.

[10] K. H. Malki, "Voice Disorders Among Saudi Teachers in Riyadh City," *Saudi Journal of Oto-Rhinolaryngology Head and Neck Surgery*, 31(3):189-199, 2012.

[11] T. Matsui, and S. Furui, "Comparison of Text-Independent Speaker Recognition Methods using VQ-Distortion and Discrete/Continuous HMMs," *Proceeding of ICASSP*, pp. 157-164, March 1992.

[12] V. Mitra, H. Nam, C. Y. Espy-Wilson, E. Saltzman, and L. Goldstein, "Articulatory Information for Noise Robust Speech Recognition," *IEEE Transactions On Audio, Speech, And Language Processing*, 19(7):1913-1924, 2011.

[13] G. Muhammad, T. A. Mesallam, K. H. Malki, M. Farahat, M. Alsulaiman, and M. Bukhari, "Formant Analysis in Dysphonic Patients and Automatic Arabic Digit Speech Recognition." *BioMedical Engineering OnLin*e, 10(40):1-12, 2011.

[14] National Institute on Deafness and Other Communication Disorders: Voice, Speech, and Language: Quick Statistics, 2014. Available at http://www.nidcd.nih.gov/health/statistics/vsl/Pages/stats.aspx. Accessed on 20 May, 2014.

[15] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 3rd Ed., Cambridge University Press, UK., 2007.

[16] Research Chair of Voicing and Swallowing Disorders. Available at http://c.ksu.edu.sa/vas/en/vsb. Acessed on 20 July, 2014.

[17] N. Roy, R. M. Merrill, S. Thibeault, R. A. Parsa, S. D. Gray, and E. M. Smith, "Prevalence of Voice Disorders in Teachers and the General Population," *J Speech Lang Hear Res.*, 47(2):281-93, Apr 2004.

[18] Y. Shao, and C. Chang, "Bayesian Separation with Sparsity Promotion in Perceptual Wavelet Domain for Speech Enhancement and Hybrid Speech Recognition," *IEEE Transactions On Systems, Man, And Cybernetics—Part A: Systems And Humans*, 41(2):284-293, 2011.

[19] C. Wu, and W. Liang, "Emotion Recognition of Affective Speech Based on Multiple Classifiers using Acoustic-Prosodic Information and Semantic Labels, *IEEE Transactions On Affective Computing*, 2(1):10-21, 2011.

Zulfiqar Ali obtained his MS degree in Computer Science from University of Engineering and Technology (UET), Pakistan with the specialization in system engineering. Since 2010, he is working as a researcher in the Speech Processing Group at the department of computer engineering, King Saud University, Saudi Arabia. He is also a member of Center for Intelligent Signal and Imaging Research (CISR), Universiti Teknologi PETRONAS (UTP), Malaysia. His research interests include Speech/speaker recognition, Automatic Voice Pathology Assessment, Computer-aided Pronunciation Training System, and Bio-medical Applications.

Ghulam Muhammad received Bachelor degree in Computer Science and Engineering from Bangladesh University of Engineering and Technology. He obtained Master degree and Ph.D. from Toyohashi University of Technology, Japan in 2003 and 2006, respectively. He is currently an Associate Professor in computer engineering department, King Saud University. His research topics include automatic speech and speaker recognition, digital signal processing, multimedia forensics, etc.

Mansour Alsulaiman is associate professor in Department of Computer Engineering at King Saud University, Riyadh, Saudi Arabia. He obtained his PhD degree from Iowa State University, USA in 1987. Since 1988, he is associated with Computer Engineering Department, King Saud University. He is editor-in-chief of King Saud University Journal Computer and Information Systems. His research areas include Automatic Speech/Speaker Recognition, Automatic Voice Pathology Assessment Systems, Computer-aided Pronunciation Training System, and Robotics.

Irraivan Elamvazuthi obtained his PhD from Department of Automatic Control & Systems Engineering, University of Sheffield, UK in 2002. He is currently an Associate Professor at the Department of Electrical and Electronic Engineering, UniversitiTeknologi PETRONAS (UTP), Malaysia. His research interests include Control, Robotics, Mechatronics, Power Systems and Bio-medical Applications.

Khalid AlMutib obtained his PhD degree in 1997 from School of Engineering and Information Sciences, University of Reading, UK. He did his MS in Electrical and Computer Engineering from University of Kansas, Lawrence Kansas, USA in 1985. Since 1997, he is working as Assistant professor at Department of Software Engineering, King Saud University, Saudi Arabia. His research areas include Robotics, and Artificial Intelligence.

LSR-Based Core Selection in Shared Tree Multicasting

Bidyut Gupta*, Sindoora Koneru*, and Shahram Rahimi*
Southern Illinois University, Carbondale, IL 62901

Abstract

Multicasting can be done in two different ways: source based tree approach and shared tree approach. In shared tree approach, a single shared tree is needed for forwarding multicast packets and in source based tree approach, construction of a minimum cost tree per source is needed for packet transfer. Hence, shared tree multicasting is preferred over source based tree approach. Several protocols like Core Based Tree (CBT), and Protocol Independent Multicasting Sparse Mode (PIM-SM) use shared tree approach. In this paper, we have presented two new concepts named pseudo sub-diameter and super pseudo sub-diameter which have been used for efficient core selection on networks that use Link State Routing protocol as the underlying unicast protocol. In fact, these two concepts are themselves independent of whatever underlying unicast protocol is being used. The proposed static core selection method uses pseudo sub-diameter concept and the group based core selection method uses super pseudo sub-diameter concept. The presented methods select more than one core to achieve fault tolerance.

Key Words: Core selection, link state routing, pseudo sub-diameter, super pseudo sub-diameter, shared tree.

1 Introduction

With the increase in the availability of internet, many applications such as video/audio broadcasting, video conferencing, and resource discovery have become popular. Most of these applications work efficiently because of the underlying reliable and efficient multicast routing protocols. Hence, choosing an effective multicast routing protocol has become critical. Multicasting is preferred over multiple unicasts from the viewpoint of better utilization of network resources, mainly network bandwidth. Hence, various multicast communication protocols have been developed since the 1980s [1, 4-5, 8, 9, 12-13] including flooding, spanning tree, reverse path forwarding, and core-based trees (CBT).

Each multicast routing protocol needs a distribution tree to send packets to the receivers. In general, multicast trees generated by multicast routing protocols can be divided into two categories, namely, source based distribution trees [8-9, 10, 12] and shared trees [1, 4-5]. Among these multicast techniques, the shared tree approaches have become more efficient compared to source based tree approaches because, unlike source based tree approaches, shared tree approaches use single shared tree for packet forwarding. The traditional CBT protocol [1] involves having a single node, known as the core of the tree, from which the branches stretch. These branches form the shortest paths between the members of the multicast group and the core.

The main disadvantages of the core based tree method are core as a single point failure and core selection. A single point failure on the tree will partition the tree and make it difficult, if not impossible, to fulfill the requirement of multicasting. And, selection of a core is NP-complete problem. While various solutions have been proposed to address these problems, they usually either require knowledge of the entire topology or are not always fault tolerant.

Authors, in [13], have proposed a method for core placement. The key idea of the proposed method has been derived from the concept that the paths between cores and members are the shortest paths. In practice there may be a number of multicast sessions going on in parallel. In such situations using a single core may become a problem. Core as a single point failure problem is addressed in this work. When load on the current core exceeds a maximum threshold, node with next maximum weight is used as a core. The major drawback of this maximum weight method is that it requires the knowledge of the entire network topology to calculate the shortest paths between nodes. In addition, the shortest path needs to be calculated between all pairs of nodes in the network. This can be exhaustive.

In [2-3], authors have proposed a multicast core management protocol for networks using Link state routing protocol (LSR). Link state core management protocol (LCM) uses a Core Binding Server (CBS) which maintains the bindings of core-group for all active groups in the network. A node that wishes to join a multicast group sends a core-mapping message to CBS. If CBS finds the corresponding core, it sends core address to the node. The node then attaches itself to the multicast tree. Otherwise, CBS selects a core and binds it to the group. In general, initially, the first node in the group becomes a core. Periodically, the core node computes a shortest-path tree to reach the members of group, and finds the center of the resulting tree. The major drawback of this

* Department of Computer Science. Email: [bidyut, rahimi]@cs.siu.edu, sindoora.koneru@gmail.com.

approach is the use of one centralized server, CBS, for the management of core-group bindings. This raises the concern of the workload overhead at the server. The Optimal Cost Based Tree (OCBT) [11, 14, 19] approach calculates the cost (number of links) of the tree rooted at each router in the network and selects the one which gives the lowest maximum delay over all other roots with lowest cost.

Problem formulation: In this work, we have extended our recent work [16] on static core selection to group based core selection. We consider networks which use Link State routing (LSR) protocol for communication. We shall first present the static core selection method based on our recently introduced concept of pseudo sub-diameter [10, 16]. After that we shall use a second new concept known as super pseudo sub-diameter [10] to design a core selection algorithm based on only group members of a given multicast group. This new concept is derived from the concept of pseudo sub-diameter. In addition to selecting the primary core we have also considered selection of additional cores to add an element of robustness. The proposed methods are also applicable to networks which use Distance Vector Routing (DVR) as unicast protocol. Besides, the proposed methods can be applied to both wide area networks (WAN) and to inter-domain multicasting (protocol independent multicasting sparse mode (PIMSM)).

We have described, first, the concept of pseudo sub-diameter in Section 2. Static core selection method [16] is presented in Section 3. In Section 4 we present the group based core selection method. In Section 5 performance has been discussed. Finally, Section 6 draws conclusions.

2 Pseudo Sub-Diameter

The routing protocols proposed in [15, 17-18] use a new concept called pseudo diameter for networks that use distance vector routing (DVR). It is defined as follows [15, 17-18]. For a source router r_i, based on its DVR table, its pseudo diameter is the maximum value among the costs to reach from source r_i to all other routers in a network. The implication of pseudo diameter is that any other router is reachable from source router r_i within the distance (i.e., cost/no. of hops) equal to the pseudo diameter of router r_i. It directly relates to the physical location of router r_i. Pseudo diameter is not the actual diameter of the network, because it depends on the location of router r_i in the network. So, different routers in a network may have different values for their respective pseudo diameters.

Theoretical findings and simulation results have been presented in [15, 17-18] to support the claim that pseudo diameter-based packet forwarding during broadcasting and multicasting improves the utilization of network bandwidth by reducing the number of duplicate packets generated. In [12], authors have extended the application of pseudo diameter to static core selection when the underlying unicast protocol is DVR. It has been shown that the concept of pseudo diameter, when used in static core selection approach, effectively determines primary, secondary, tertiary cores etc. by only one time broadcast of pseudo diameter. The selection is unanimous because all routers have the same information needed for selection and the approach is fault tolerant as secondary and tertiary cores can be determined along with the primary core.

In this paper, to make the concept of pseudo diameter protocol independent, i.e., independent of the underlying unicast protocol, we have extended the idea to networks based on LSR. For LSR, we use pseudo sub-diameter as the equivalent term for pseudo diameter. It is seen in [15, 17-18] that, to obtain pseudo diameter, each entry in a DVR table must appear as <destination router, next hop router, cost to reach destination>. Hence, in order to acquire pseudo sub-diameter and for the core selection method to work based on this idea, corresponding unicast LSR table entries must also appear as a tuple of length 3 as in DVR tables. In Link State Routing, each entry in the Link State Table (LSRx) of any router X appears as <destination router, next hop router>. 'Cost to reach destination' is absent in the entries. However it is simple for any router X to include it, because any way router X has to determine the least cost path to every other destination router Y using topological information; that is, eventually X computes such costs. Therefore, if 'cost to reach destination' is included in LSRx, the method proposed in [10] can be applied for static core selection in LSR based networks. We have modified LSR tables by adding the 'cost to reach destination' field and thus the LSR table entries appear as a tuple of length 3. We define pseudo sub-diameter derived from the modified LSR tables as follows.

Let n be the number of routers in a network with diameter d. Let LSR_i be the unicast link state routing table used by router r_i, $[1 \leq i \leq n]$ and let $<r_j, r_k, c_{i,j}>$ be the entry for destination r_j, $[1 \leq j \leq n; j \neq i]$, and r_k be the next hop router along the shortest path from r_i to r_j with cost $c_{i,j}$. We define pseudo sub-diameter $P_d(r_i)$ of router r_i as follows:

$$P_d(r_i) = \max \{c_{i,j}\}, \text{ where } c_{i,j} = \text{cost}(r_i, r_j), c_{i,j} \in LSR_i$$

In words, for a source router r_i, based on its LSR_i table, its pseudo sub-diameter is the maximum value among the costs to reach from source r_i to all other routers in a network. Like pseudo diameter, the implication of pseudo sub-diameter is that any other router is reachable from a source router r_i within the distance (i.e. cost/no. of hops) equal to the pseudo sub-diameter of router r_i.

We summarize the characteristics of pseudo sub-diameter below.

(1) Pseudo sub-diameter of a router is directly related to the physical location of the router in a network.
(2) So it may be different for different routers.
(3) Pseudo sub-diameter of a router is not the diameter of a network.
(4) It is always less than or equal to the diameter of a network.

3 Static Core Selection in LSR – Based Networks

We have introduced a systematic approach [10, 16]to select a static core in networks which use LSR for message

communication. This core selection is independent of any multicast group and will involve all routers in a network. The working principle of the method is stated below.

Let cost $c_{i,j}$ between two routers r_i and r_j be measured in no. of hops. Let r_i be any router in a network of n routers and let T_i denote the minimal spanning tree rooted at r_i and T_i (L) be its number of levels.

Therefore,

$$T_i (L) = P_d(r_i) = \text{maximum } (c_{i,j}),\ 1 \le j \le n,\ j \ne i,\ c_{i,j} \in LSR_i$$

Hence, a router r_k will be selected as the primary static core,

if $T_k (L) = \min \{T_j (L)\}, \forall j$

i.e., $P_d(r_k) = \min \{P_d(r_j)\}, \forall j$

Note that even if the cost is measured differently, a router r_k will be selected as the primary static core,

if $P_d(r_k) = \min \{P_d(r_j)\}, \forall j$.

To incorporate core redundancy, a router r_m is selected as the secondary core,

if $P_d(r_m) = \min \{P_d(r_j)\}, j \ne k$

Similarly a tertiary core (with the next lowest P_d value) can be selected to achieve an even higher degree of fault tolerance.

Note that during core selection in the event of a tie between routers, the router with highest IP address is selected as the core of choice.

3.1 Algorithm Description

The following notations and data structures are used in the algorithm.

n: total number of routers in a network;

broad_message ($P_d(r_i)$, r_i): broadcast message from router r_i with its pseudo sub-diameter P_d;

LSRi[][]: LSR table of router r_i, $[1 \le i \le n]$;

core[][]: two dimensional array that contains router ids and corresponding pseudo sub-diameters in ascending order of pseudo sub-diameter;

Procedure broad() is called by each r_i where $(1 \le i \le n)$ to broadcast $P_d(r_i)$. Each router controls the broadcast with its P_d value, i.e., a router broadcasts a message to its neighbor only if it is within its P_d range.

Procedure receive_Broad($P_d(r_i)$, r_i) is used by each router. This procedure has $P_d(r_i)$ and r_i as input and generates core[][] that contains router ids and their respective pseudo sub-diameter values.

Procedure Static_core() called by r_i, has core[][] as input. It sorts core[][] in ascending order of the P_d values. If multiple routers have the same P_d value, the routers are sorted in the descending order of their IP addresses. At the end of this procedure, core[][] contains, in order, primary core, secondary core, tertiary core, etc., and the later ones are selected to achieve fault-tolerance.

Algorithm Static_Core

Procedure broad()

```
Begin
    for each ri [1 ≤ i ≤ n]
        Pd(ri) = max (LSRi[j][2]) [1 ≤ j ≤ n]
        send broad_message (Pd(ri), ri)
  end for
/* each node broadcasts its pseudo sub-diameter value (Pd
    based broadcast)*/
End
```

Procedure receive_Broad($P_d(r_i)$, r_i)

```
Begin
  for each ri [1 ≤ i ≤ n]
    for k = 1 to n
      if i ≠ k
        core[k][0]=rk
        core[k][1]=Pd(rk)
      end if
    end for
  end for
/*core[][], a two-dimensional array that contains router id
    and pseudo sub-diameter*/
  Static_Core(core[][])
End
```

Procedure Static_Core(core[][])

```
Begin
        sort_asc(core[][])
/*sort core[][] in ascending order of Pd*/
        if same Pd
          sort_desc(core[][])
/*sort core in descending order of router's IP address*/
        end if
    end for
End
```

3.2 An Example

In static core approach, the selected core is the router that has minimum P_d (r_i) compared to all other routers in the network. Obtaining static core requires additional information about the network, like the next hop and the cost to reach the next hop. This information is present in link state routing tables of each router. Choosing a core, based on topology, is attractive and easy as it does not require group member information or source information.

Consider the network in Figure 1. It shows an 8 router network. Let C, D, G, H be the group members of a multicast group g. Each router in the network initializes the creation of link state routing tables by sending link state advertisements.

After the exchange of Link State Advertisement packets, each router in the network has a summary routing table that is shown in Figure 2. Modified Link State routing tables are derived from this summary table. Figure 3 shows the modified LSR tables of the group members from the network provided in Figure 1. These tables contain an additional cost field which is not present in the regular LSR tables.

251660288251659264

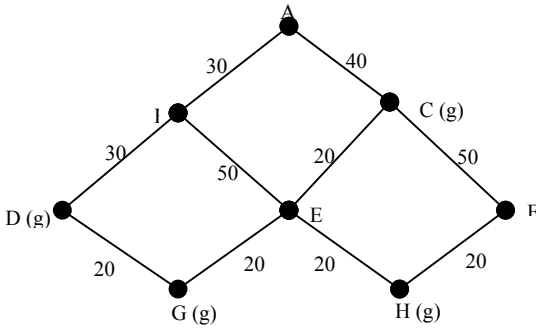


Figure 1: An 8 node network

Router	Next Hop	Cost
A	B	30
A	C	40
B	A	30
B	D	30
B	E	50
C	A	40
C	E	20
C	F	50
D	B	30
D	G	20
E	B	50
E	C	20
E	G	20
E	H	20
F	C	50
F	H	20
G	D	20
G	E	20
H	E	20
H	F	20

Figure 2: Routing table with LSAs

A		
Dest.	Next	Cost
A	A	0
B	B	30
C	C	40
D	B	60
E	C	60
F	C	90
G	C	80
H	C	80

B		
Dest.	Next	Cost
A	A	30
B	B	0
C	A	70
D	D	30
E	E	50
F	E	90
G	D	50
H	E	70

C		
Dest.	Next	Cost
A	A	40
B	A	70
C	C	0
D	E	60
E	E	20
F	F	50
G	E	40
H	E	40

D		
Dest.	Next	Cost
A	B	60
B	B	30
C	G	60
D	D	0
E	G	40
F	G	80
G	G	20
H	G	60

E		
Dest.	Next	Cost
A	C	60
B	B	50
C	C	20
D	G	40
E	E	0
F	H	40
G	G	20
H	H	20

F		
Dest.	Next	Cost
A	C	90
B	H	90
C	C	50
D	H	80
E	H	40
F	F	0
G	H	60
H	H	20

G		
Dest.	Next	Cost
A	E	80
B	D	50
C	E	40
D	D	20
E	E	20
F	E	60
G	G	0
H	E	40

H		
Dest.	Next	Cost
A	E	80
B	E	70
C	E	40
D	E	60
E	E	20
F	F	20
G	E	40
H	H	0

Figure 3: Modified LSR tables

Static core, in our approach, is a router that has least pseudo sub-diameter value compared to all other routers in the network. According to its definition, pseudo sub-diameter of a router is the maximum cost present in its LSR table. So, from the modified LSR tables, each router in the network can derive its pseudo sub-diameter value. From Figure 3, pseudo sub-diameter of routers A, B, C, D, E, F, G, and H are obtained as 90, 90, 70, 80, 60, 90, 80, and 80, respectively.

Each router broadcasts its $P_d(r_i)$ value to all other routers in the network (r_i, $1 \leq i \leq n$). At the end of the broadcast, each router contains P_d of every other router in the network as shown in Figure 4a. Observe in Figure 4a that there is more than one router with the same P_d value. For example, routers D, G, and H have the same P_d value of 80. If this situation arises, the routers are sorted in descending order of their router ids (IP addresses). Router H has the highest IP address, followed by routers G and D respectively. Router r_i calls Procedure Static_Core (core[][]) which returns sorted core[][] array with primary core, secondary core, tertiary core etc. in

order as shown in Figure 4b. According to Figure 4b, router E is the primary static core, as it is the one with the least pseudo sub-diameter value of 60. That is, the maximum # hops (cost) to reach any router from this core is its P_d value which is 60 and since this P_d is less than the pseudo sub-diameters of all other routers, it is the best choice. Similarly, to maintain fault tolerance, router C and H can be chosen as the secondary and tertiary cores respectively as they are the ones with next least pseudo sub-diameter value. Note that H has been selected as the tertiary core because it has the highest IP address among the routers D, G, and H. Observe that the information present in the LSR tables is sufficient for selecting the core.

r_i	$P_d(r_i)$
A	90
B	90
C	70
D	80
E	60
F	90
G	80
H	80

Figure 4a: Content in a route r_i after broadcast

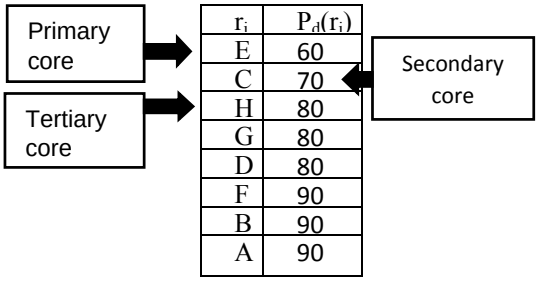


Figure4b: Static cores selected by any router r_i

4 Super Pseudo Sub-Diameter

Super pseudo sub-diameter of a router r_i, denoted as $P_d^s(r_i)$ is defined as follows.

$$P_d^s(r_i) = \max \{c_{i,j}\}, \quad \text{where} \quad c_{i,j} = \text{cost}(r_i, r_j), \; r_j \in g \text{ and } c_{i,j} \in LSR_i$$

In words, for a source router r_i, based on its unicast LSR table, LSR_i, its super pseudo sub-diameter denoted as $P_d^s(r_i)$ is the maximum value among the costs to reach from source r_i only those routers which are connected to group members of a given multicast group g. Therefore, like pseudo sub-diameter, super pseudo sub-diameter of any router r_i also directly relates to the physical location of the router and we have used this interpretation of such diameter to determine a group based core. The important point to note is that the information present in the LSR tables is sufficient for selecting the core.

The information about costs is collected from the source router's LSR routing table. The portion of LSR_i containing only the routers attached to group members along with the respective next hops / cost-information is termed as the sub-LSR table of router r_i and is denoted as sub-LSR_i. Therefore, sub-$LSR_i \subseteq LSR_i$. Observe that super pseudo sub-diameter of a router is always less than or equal to its pseudo sub-diameter, since the router computes the value of its super pseudo sub-diameter from a subset of the routers (which are connected to group members) in its LSR routing table. Hence, it is very likely that super pseudo sub-diameter will be less than the network diameter most of the time.

Theorem 1: $\forall i, P_d(r_i) \geq P_d^S(r_i)$

Proof follows directly from the above discussion.

Static core can co-exist with group based core mechanisms. Group based core takes precedence over statically configured cores [16]. Group membership information needs to be known in order to select a core and the chosen core is to be a group member. In certain scenarios, based on the distribution of group members, group member based core may be the same as the static core. Below we informally state the working principle of the group based core selection approach and explain how it works with an example. Later, we formally present the group based core selection algorithm.

4.1 Working Principle

Step 1. Each router, r_i connected to a group member sends its membership-query request with its id (IP address) to static core, say, x.
Step 2. x forwards all received ids to each r_i.
Step 3. If x is a group member
  Step 3.1. x derives its sub-LSR table, say, sub-LSR_x
  Step 3.2. x obtains its super pseudo sub-diameter, $P_d^S(x)$ from its sub-LSR_x table.
Step 4. Each r_i derives its sub-LSR_i, containing ids of all routers connected to group members.
Step 5. Each r_i obtains its super pseudo sub-diameter, $P_d^S(r_i)$
Step 6. Each r_i unicasts $P_d^S(r_i)$ to x.
Step 7. x identifies the router r_k with lowest $P_d^S(r_k)$ as the primary core.
  Next lowest one is the secondary core, and similarly the following one is the tertiary core.
  In case of a tie, x prioritizes the one with highest id.
Step 8. Static core x forwards the ids of primary core, secondary core, and tertiary core to all group members.

An Example: In group core based shared tree approach, the router that is chosen as core is a group member. Consider the network in Figure 1. Let C, D, G, and H be the routers connected to group members of a given multicast group denoted as 'g' in the diagram. Each such router sends a membership-query request with its id to the static core E. Static core E, after receiving all membership requests from the group members generates a data structure, which contains group member ids. If static core is a group member, it derives its sub_LSR table and obtains its super pseudo sub-diameter. Static core sends the data structure with ids of group members to all group members. Figure 5a shows the initial data

structure that static core E sends to the group members. Each group member, upon receiving the information from static core, derives its own sub-LSR table. It then obtains its P_d^S value. Figure 5b shows sub-LSR tables of each group member. Their P_d^S values are highlighted. Each group member then unicasts this information to the static core. Static core, upon receiving P_d^S values from all the group members, updates the data structure with corresponding super pseudo sub-diameter values. It then sorts the data structure in the ascending order of the P_d^S values. If there are multiple routers with the same P_d^S value, data structure is sorted in descending order of routers' IP addresses. Figure 5c shows the final sorted data structure. So, according to Figure 5c, G is the primary core as it has the lowest P_d^s value. Since C, D, and H have the same P_d^s value, assuming H has the highest IP address among them followed by D and C, we consider H as the secondary core. Similarly, D is chosen as the tertiary core. E forwards this information to all the group members. Now each group member has the same information about the primary group core, secondary group core and tertiary group core. Thus, the group based core is determined considering only the routers connected to group members.

Group id
C
D
G
H

Figure 5a: Initial data structure from static core containing group Ids

C		
Dest.	Next	Cost
D	E	**60**
G	E	40
H	E	40

D		
Dest.	Next	Cost
C	G	**60**
G	G	20
H	G	60

G		
Dest.	Next	Cost
C	E	**40**
D	D	20
H	E	40

H		
Dest.	Next	Cost
C	E	40
D	E	**60**
G	E	60

Figure 5b: Sub-LSR tables of group members

Group id	P_d^S
G	40
H	60
D	60
C	60

Figure 5c: Final data structure from static core containing sorted group cores information

4.2 Algorithm Description

The following notation and data structures are used in the algorithm.

SC: static core;
G: a multicast group;
r_p: a router ϵ G;
unicast (SC, r_p): unicast message from group member r_p to static core, SC;
core[][]: an array with group members ids and their corresponding super pseudo sub-diameter values;
core(r_p)[][]: sub-LSR table of group member r_p;
core(SP)[][]: sub-LSR table of static core;
$P_d^S(r_p)$: super pseudo sub-diameter of group members r_p

Procedure unicast() is called by every router r_p (ϵ G). Each r_p unicasts router ids to the static core.
Procedure SC_SubLSR (r_p) is called by static core, SC, which has r_p as input. If SC ϵ G, it generates sub_LSR table, core(SP)[][], and core[][] with group ids. Otherwise, it generates core[][]. SC forwards core[][] to all group members.
Procedure GM_SubLSR (r_p) is called by every router r_p that ϵ G upon receiving core[][] from static core, SC. This procedure generates super pseudo sub-diameter of each group member, $P_d^S(r_p)$ as output from its corresponding sub_LSR table, core(rP)[][]. Each group member then sends its $P_d^S(r_p)$ to the static core, SC.
Procedure Group_Core (r_p, $P_d^S(r_p)$) is executed by the static core, SC. It has router id and corresponding super pseudo sub-diameter value of each group member as input and generates core[][] containing primary Group-core, secondary group-core, tertiary group-core, etc., as output, to provide fault tolerance.

Algorithm Group_Core
Procedure unicast()
Begin
for each r_p ϵ G
sends unicast(SC, r_p)
end for
/* all group members send their id to static core*/
End

Procedure SC_SubLSR(r_p)
Begin
$\forall$ r_p ϵ G
receives unicast(r_p)
if SC ϵ G
derives core(SP)[][]
generates core[][]
end if
else
generates core[][]
end else
/*if SC ϵ G, it derives sub-LSR tables and core[][] with group member ids; otherwise SC derives core[][]*/
$\forall r_p \epsilon$ G

sends core[][]
/* Static core, SC sends group membership information to all the group members*/
End

Procedure GM_SubLSR(r_p)
Begin
for each $r_p \in G$
receives core[][]
derives core(r_p)[][]
obtains $P_d^s(r_p)$
/*derives sub-LSR tables and obtains its super pseudo sub-diameter value*/
unicasts (r_p, $P_d^S(r_p)$) to SC
/*group members send their $P_d^S(r_p)$ value to the Static core, SC */
End

Procedure Group_Core(r_p, $P_d^S(r_p)$)
Begin
for each $r_p \in G$
receives (r_p, $P_d^S(r_p)$)
updates core[][] with $P_d^S(r_p)$
end for
/*adds super pseudo sub-diameter values of group members to core[][] */
sort_asc(core[][])
/*sort core[][] in ascending order of P_d^S */
if same P_d^S
sort_desc(core[][])
/*sort core in descending order of router's IP address*/
end if
$\forall r_p \in G$
sends core[][]
/*Static Core, SC, sends core[][]to all group members */
End

In our approach each group member has the same information about primary, secondary, and tertiary cores. This helps in selecting new core when there is primary core failure. We assume that these cores periodically exchange hello messages.

Consideration of Primary Core failure. Core replacement during Primary core failure follows the steps stated below.

- If secondary core does not hear from primary core during one such interval, it unicasts primary core failure information to all group members.
- Primary core is deleted from the final data structure. Next entry in the order is chosen as secondary core. This choice is unanimous because all routers have the same final data structure.

In a similar way, if the secondary core fails while it is working as a primary core, the tertiary core can take care of the primary core's responsibility. In this context, it is worth mentioning that in the event of the primary core leaving the group, it must send a group_leave() message to inform all other group members about its leaving.

5 Performance

It is clear that the complexity of static core based approach is $O(n^2)$ since each router sends its P_d to all other routers; n being the total number of routers in the network. Similarly, for group based approach, the message complexity is $O(m)$ where m is the number of group members in the network. Both static core selection method and group based core selection method are compared with some important core selection methods.

We now briefly state the complexities of each of these methods. Maximum path count core selection method [6] finds the shortest paths for all pairs of nodes in the given network. The nodes are then sorted in descending order of their path counts. The first nodes are selected to be the candidate cores. The complexity of this approach is $O(n^2)$ where n is the number of nodes in the network. Minimum average distance method [6] finds the average distance along the shortest paths from each node to all other nodes in the network. The nodes are sorted in ascending order of their average distance. The first nodes are selected to be the candidate cores. The complexity of this approach is $O(n^2)$, where n is the total number of nodes in the network.

In Delay Variant Multicast Algorithm (DVMA) [7] it is assumed that the complete topology is available at each node (router). The algorithm starts with a spanning tree satisfying the delay constraint only, which may not include some destination nodes. Then the algorithm searches through the candidate paths satisfying the delay and delay-variation constraints from a non-tree member node to any of the tree nodes. It works on the principle of k-shortest paths to the group of destination nodes concerned. If these paths do not satisfy a delay constraint, then it may find a longer path, which is a shortfall of DVMA. The spanning tree built by DVMA satisfies both delay and delay-variation constraints. However, due to the very high time complexity, it does not fit in a modern high-speed computer network environment. The worst case complexity of DVMA is $O(kldn^4)$, where k and l are the number of paths satisfying the delay bound between any two nodes, $|D| = d$ and $|N| = n$ represents the number of multicast receptor nodes and total number of nodes in the topology network, respectively.

Tournament [3, 7] core selection method operates in four stages. During the first level of tournament, the process involves the comparison of distances between the node pairs for the matching process, and takes square time in the group size, i.e., $O(|M|^2)$, where M is the number of multicast group members. Triggering the pair leader requires transmission of pair knowledge to it from the coordinator. The number of messages transmitted in Step 2 is half the number of candidates at current tournament level and hence $O(|M|)$. The process of locating the node closest in cost-distance to the midpoint of a pair in Step 3 merely requires the knowledge of nodes that lie on the shortest path connecting the pair available

to the pair leader, requiring transmission of $O(|V|)$ messages, where V is the number of nodes in the network, and comparison of node distances of the intermediary nodes is done in $O(|V|)$ time at the pair leader. In Step 4, as many messages as in part (2) are transmitted back to the coordinator. Throughout the tournament, the steps (1 to 4) are iterated in n levels where $n = \log_2(|M|)$. Since the matching process at coordinator operates in time $O(|M|^2)$ at the first level and $O(|M|)$ at subsequent levels, the overall complexity of the tournament coordinator is, therefore, $O(|M|^2 + |M| + \log_2(|M|)) = O(|M|^2)$.

OptTree [7] method suggests optimization criteria which can be computed for one core in the iteration that the core is examined as a candidate, without changing the structure of their proposed greedy algorithm [7]. If we consider a combination of Node-count, Delay-residue, aveHop and Degree as the optimization criterion of the algorithm, the additional complexity to the greedy algorithm is clearly that of the one among four with the highest complexity. In Node-count, Delay-residue and aveHop of optTree, each candidate core is examined for its rank depending on the receiver set they are potentially dominating—repeating at most as many times as the number of receivers. Delay-residue and aveHop also examine the distances of the core from the sources for residue in delay and hop-distances to the receivers. Therefore, the complexity of optTree is $O(|M|^3|C|)$ where M is the number of multicast group members and C is the number of candidate cores.

Topology Setup. In our experimental setup we have used the NS2 simulator. BRITE topology generator is used to create flat random graphs. In BRITE topology, we have chosen the Waxman model for topology generation which uses Waxman's probability model for interconnecting the nodes of the topology. From the provided heavy-tailed and random approach for node placement, we have used random placement approach in which each node is placed in a randomly selected location of the plane. BRITE offers the choice of degree of connectivity for the nodes. Without any loss of generality we have arbitrarily chosen 4 links per node and nodes are added to the smallest degree non-leaf node to form the network. Nodes are connected using duplex links. BRITE assigns bandwidths to links according to one of four possible distributions, namely constant, uniform, exponential, and heavy-tailed. BRITE also offers to choose arbitrarily the values of minimum and maximum bandwidth; for example, they can be 1Kbps and 10 Mbps, respectively. Without any loss of generality we have chosen the above mentioned values and used uniform bandwidth assignment on the links where a value is distributed uniformly between minimum and maximum bandwidth values. We have used UDP to send packets of size ranging from 500 bytes to 1 Kbyte randomly at a Constant Bit Rate (CBR) of 800 Kbps.

Topology Generation. We have experimented with 10 different 30 router topologies, 40 router topologies, and 50 router topologies. In each topology group size varies from 5 to 25. For each 30 router topology multiple routers are chosen as sources for each group. We have measured the total number of packets generated for core selection in our approach. We have then compared it with some existing important related approaches. Figure 6 shows the average number of packets generated on all 30 router topologies for group based core selection approaches considered in this work as well as for the methods with which we are comparing. In a similar way average numbers of packets generated for 40 and 50 router network topologies are shown in Figures 7 and 8, respectively. Simulation results have shown the superiority of our approach to these other approaches.

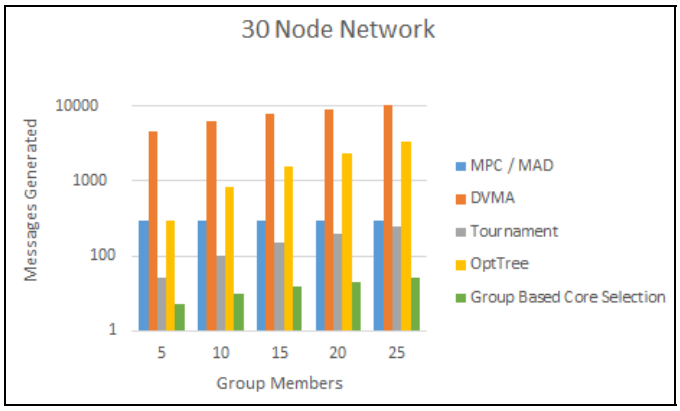


Figure 6: 30 node network

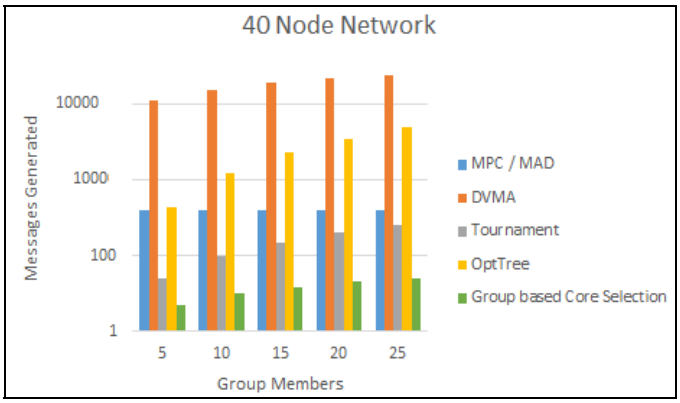


Figure 7: 40 node network

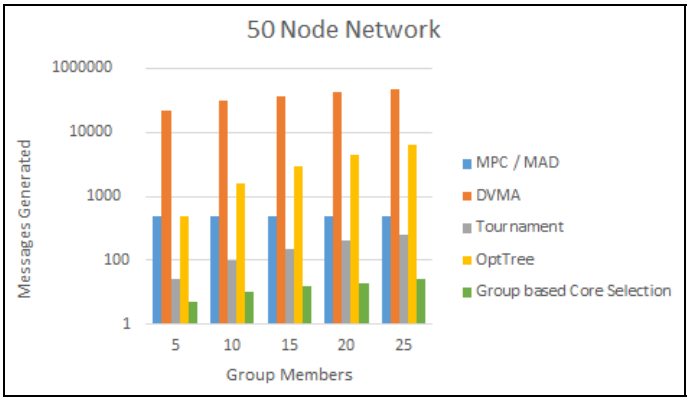


Figure 8: 50 node network

6 Conclusion

In this paper we have presented novel approaches for core selection for networks that use LSR as the unicast routing protocol. Both the core selection approaches have been made fault tolerant, i.e., in addition to primary core the presented methods also determine a secondary, tertiary core etc. to achieve fault tolerance. The most noteworthy point of the presented work is that the routing information present in the LSR tables is sufficient for core selection, be it static core or group-based core. We have given a new interpretation of the information present in the LSR tables, which have resulted in the two new concepts, namely, pseudo sub-diameter and super pseudo sub-diameter. These two concepts have been used to design the core selection approaches. The importance of these two concepts lies in the fact that these are directly related to the physical locations of the routers; therefore it justifies their use in the core selection approaches. In fact, these two concepts are themselves independent of whatever underlying unicast protocol is being used. The proposed methods can also be applied in PIMSM for inter-domain multicasting as well as multicasting in WANs. The simulation results show the superiority of our proposed core selection approaches to some related important existing approaches from the viewpoint of lower message complexity.

References

[1] Andrew Adams, Jonathan Nicholas, and William Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM)," Internet Engineering Task Force (IETF),RFC-3973, January 2005.

[2] Tony A. Ballardie, Paul Tsuchiya, and Jon Crowcroft, "Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing," ACM SIGCOMM 93, San Francisco, September 1993.

[3] Stephen E. Deering and David R. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Transactions on Computer Systems (TOCS)*, 8(2):85-110, May 1990.

[4] Bill Fenner, Mark Handley, Hugh Holbrook, and Isidor Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM), " Internet Engineering Task Force (IETF),RFC-4601, August 2006.

[5] Bidyut Gupta, Sindoora Koneru, and Shahram Rahimi, "LSR-Based Static Core Selection in Shared Tree Multicasting," *Proceedings of CAINE*, 2014, pp. 181-186, New Orleans, Oct. 2014.

[6] Y. Huang, E. Fleury, and P. K. McKinley, "LCM: A Multicast Core Management Protocol for Link-State Routing Networks," *IEEE International Conference on Communications*, 2:1197-1201, June 1998.

[7] Y. Huang and P. K. McKinley, "Group Leader Election under Link-State Routing," *Proceedings of International Conference on Network Protocols*, pp. 95-104, 1997.

[8] A. Karaman and H. Hassanein, "Core-Selection Algorithms in Multicast Routing—Comparative and Complexity Analysis," *Journal of Computer Communications*, 29(8):998-1014, May 2006.

[9] Sindoora Koneru, Bidyut Gupta, Shahram Rahimi, and Ziping Liu, "Hierarchical Pruning to Improve Bandwidth Utilization of RPF-Based Broadcasting," *IEEE Symposium on Computers and Communications (ISCC),* Split, Croatia, pp. 96-100, July 5-7, 2013.

[10] Sindoora Koneru, Bidyut Gupta, Narayan Debnath, "A Novel DVR Based Multicast Routing Protocol with Hierarchical Pruning," *International Journal of Computers and Their Applications, ISCA*, 20(3):184-191, September 2013.

[11] Sindoora Koneru, Bidyut Gupta, Shahram Rahimi, Ziping Liu, Narayan Debnath, "A Highly Efficient RPF-Based Broadcast Protocol Using a New Two-level Pruning Mechanism," *Journal of Computational Science (JOCS),* Elsevier Publications, 5(3):645-652, March 2014.

[12] Sindoora Koneru, *Pseudo Diameter - A Novel Concept in Designing Highly Bandwidth Efficient Multicast Routing Protocols*, PhD Dissertation, Department of Computer Science, Southern Illinois University, 2014.

[13] H. C. Lin and Z. H. Lin, "Selection of Candidate Cores for Core-Based Multicast Routing Architectures," *IEEE International Conf. on Communalizations*, 4:2662-2666, 2002.

[14] Hwa-Chun Lin and Shou-Chuan Lai, "A Simple And Effective Core Placement Method for The Core Based Tree Multicast Routing Architecture," *Proceedings of the IEEE International on Performance, Computing, and Communications*, pp. 215-219, February 2000.

[15] George N. Rouskas and Ilia Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Constraints," *IEEE Journal on Selected Areas in Communications*, 15(3):346-356, April 1997.

[16] Young-Chul Shim and Shin-Kyu Kang, "New Center Location Algorithms for Shared Multicast Trees," *NETWORKING 2002: Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications, Lecture Notes in Computer Science, SpringerLink*, Berlin, Heidelberg, 2345:1045-1056, 2002.

[17] David G. Thaler and Chinya V. Ravishankar, "Distributed Center-Location Algorithms," *IEEE Journal on Selected Areas in Communication*, 15(3):291-303, April 1997.

[18] David Waitzman, Craig Partridge, and Stephen E. Deering, "Distance Vector Multicast Routing Protocol (DVMRP)," Internet Engineering Task Force (IETF), RFC 1075, November 1988.

[19] L. Wei and D. Estrin, *A Comparison of Multicast Trees and Algorithms*, Tech. Rep., USC-CS-93-560, University of South Carolina, Sep. 1993.

Bidyut Gupta is a Professor of Computer Science at Southern Illinois University. His current research interests include distributed systems, cluster computing, mobile computing, and computer networks. Dr. Gupta is a senior member of IEEE and a senior member of ISCA.

Sindoora Koneru received her Ph.D. from the Department of Computer Science at Southern Illinois University in 2014. Her current area of research is focused on developing, enhancing, analyzing and implementing routing protocols of wide area networks.

Shahram Rahimi is a Professor of Computer Science at Southern Illinois University. His current research interests include computational intelligence, soft computing, expert systems, computing with words, and software agents. Dr. Rahimi is an Editor in Chief of International Journal of Computational Intelligence Theory and Practice.

A Node-to-Set Disjoint Paths Routing Algorithm in Torus-Connected Cycles*

Antoine Bossard†
Tokyo Metropolitan University, Shinagawa, Tokyo 140-0011, JAPAN

Keiichi Kaneko‡
Tokyo University of Agriculture and Technology, Koganei, Tokyo 184-8588, JAPAN

Abstract

Parallel processing and distributed computing are being actively researched, and multiple topologies have been described in the literature as the interconnection network of all the processors involved in such parallel systems. Considering the huge number of nodes involved in modern multicomputers, such network topologies frequently rely on hierarchical structures, being in this case called Hierarchical Interconnection Networks (HIN). We have recently introduced the Torus-Connected Cycles (TCC) network topology which is such a HIN since it realizes the combination of an n-dimensional torus and $2n$-cycles, arranged into two separate layers, cycles forming clusters. We propose in this paper a node-to-set disjoint paths routing algorithm in a TCC, and formally show its correctness and performance through complexity analysis.

Key Words: Multicomputer; interconnection network; parallel processing; fault-tolerant routing; performance evaluation.

1 Introduction

Parallel processing is an increasingly hot topic. Effectively, due to its numerous applications and continuous advances on the hardware front such as VLSI, modern supercomputers such as the Fujitsu K [19] are massively parallel systems. Because these systems include hundreds of thousands of computing nodes (CPU), node interconnection is a critical topic, and several network topologies have been proposed as interconnection networks [4, 8, 16, 17, 18, 24]. It is worth noticing that most of the topologies proposed for interconnection networks of modern parallel systems are now hierarchical structures; they are hierarchical interconnection networks (HIN).

In this paper, we describe a node-to-set disjoint paths routing algorithm in a k-ary n-dimensional torus-connected cycles network $TCC(k,n)$. Concretely, in a $TCC(k,n)$, given a node source $\boldsymbol{s}$ and a set of three destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}$, the proposed algorithm generates three mutually node-disjoint (simply "disjoint" hereinafter) paths (at the exception of the node $\boldsymbol{s}$ which is shared amongst the paths) connecting $\boldsymbol{s}$ and $\boldsymbol{d}_i$ ($1 \leq i \leq 3$). We then formally show the algorithm correctness and performance through complexity analysis.

Node-to-set disjoint paths routing is one problem amongst other disjoint paths routing problems. For instance, node-to-node disjoint paths routing (a.k.a. the container problem) and set-to-set disjoint paths routing are two related routing problems. The former is about generating disjoint paths between two distinct nodes, while the latter is about generating disjoint paths between multiple source nodes and multiple destination nodes. A variant of the latter fixes the source-destination node pairs to be connected; this is k-pairwise disjoint paths routing. Disjoint paths routing is indeed an important issue with respect to parallel systems inter-node communication, which explains the numerous related research works.

We detail a few of these previous works. The container problem has been solved in a hypercube [22], a pancake graph [23], a burnt pancake graph [14], and a torus-connected cycles network [6]. The node-to-set disjoint paths routing problem has been solved in a hypercube [21], a torus [10], a star graph [12], a pancake graph [15], a burnt pancake graph [13], a perfect hierarchical hypercube [1], and a hierarchical cubic network [3]. The set-to-set disjoint paths routing problem has been solved in a hypercube [9], a star graph [11], a pancake graph [20], a perfect hierarchical hypercube [2], and a hierarchical cubic network [5].

The rest of this paper is organized as follows: we first recall in Section 2 definitions and notations. In Section 3, we describe a node-to-set disjoint paths routing algorithm in a TCC network. The correctness and performance of this algorithm are formally established in Section 4. Finally, this paper is concluded in Section 5.

*An extended abstract of this paper has been published in [7].
†Advanced Institute of Industrial Technology. Email: abossard@aiit.ac.jp.
‡Graduate School of Engineering.

2 Preliminaries

In this section, we begin by recalling the definition of the torus-connected cycles HIN, before giving several notations and previous results used throughout this paper.

Definition 1. A k-ary n-dimensional torus-connected cycles network $TCC(k,n)$ is an undirected graph that has $2nk^n$ nodes. Each node $\boldsymbol{a}$ has a cluster ID $\boldsymbol{c}(\boldsymbol{a}) = (a_0, a_1, \ldots, a_{n-1})$ and a processor ID $p(\boldsymbol{a}) = p_{\boldsymbol{a}}$, and the node consists of the pair $(\boldsymbol{c}(\boldsymbol{a}), p(\boldsymbol{a}))$ where $0 \leq a_i \leq k-1$ and $0 \leq p_{\boldsymbol{a}} \leq 2n-1$. Each node $\boldsymbol{a}$ has three neighbor nodes $\boldsymbol{n}_1(\boldsymbol{a})$, $\boldsymbol{n}_2(\boldsymbol{a})$ and $\boldsymbol{n}_3(\boldsymbol{a})$:

$$\boldsymbol{n}_1(\boldsymbol{a}) = (\boldsymbol{c}(\boldsymbol{a}), (p_{\boldsymbol{a}} + (-1)^{p_{\boldsymbol{a}}}) \mod 2n)$$
$$\boldsymbol{n}_2(\boldsymbol{a}) = (\boldsymbol{c}(\boldsymbol{a}), (p_{\boldsymbol{a}} - (-1)^{p_{\boldsymbol{a}}}) \mod 2n)$$
$$\boldsymbol{n}_3(\boldsymbol{a}) = (a_0, a_1, \ldots, (a_{\lfloor p_{\boldsymbol{a}}/2 \rfloor} + (-1)^{p_{\boldsymbol{a}}}) \mod k, \ldots, a_{n-1}, p_{\boldsymbol{a}} + (-1)^{p_{\boldsymbol{a}}})$$

If $n = 1$, each node $\boldsymbol{a}$ has degree two as $\boldsymbol{n}_1(\boldsymbol{a}) = \boldsymbol{n}_2(\boldsymbol{a})$. If $n \geq 2$, each node has degree three. For any node $\boldsymbol{a}$, the edges in Definition 1 $(\boldsymbol{a}, \boldsymbol{n}_1(\boldsymbol{a}))$ and $(\boldsymbol{a}, \boldsymbol{n}_2(\boldsymbol{a}))$ are called *internal*, while the edge $(\boldsymbol{a}, \boldsymbol{n}_3(\boldsymbol{a}))$ is called *external*. If all the external edges from a $TCC(k,n)$ are removed, the remaining graph consists of n^k disjoint cycles. Each cycle consists of $2n$ nodes. The cycle (a.k.a. cluster) that includes a node $\boldsymbol{a}$ is denoted by $C(\boldsymbol{a})$. Also, if each of the n^k cycles of a $TCC(k,n)$ is contracted into a single node, a k-ary n-dimensional torus (a.k.a. (k,n)-torus) is obtained.

Definition 2. Two clusters C_1 and C_2 of a $TCC(k,n)$ are *adjacent* if and only if there exists an external edge between them. The set of the adjacent clusters of a cluster C is denoted by $N(C)$.

As an example, a 3-ary 2-dimensional torus-connected cycles network $TCC(3,2)$ is given in Figure 1. For instance, the node (0, 0, 3) is adjacent to the two nodes (0, 0, 0) and (0, 0, 2) by internal edges, and adjacent to the node (0, 2, 2) by an external edge.

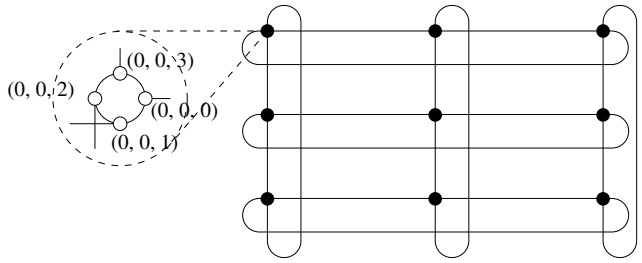


Figure 1: A 3-ary 2-dimensional torus-connected cycles network $TCC(3,2)$

We give a few additional definitions. In a graph, a path is an alternate sequence of nodes and edges $\boldsymbol{a}_1, (\boldsymbol{a}_1, \boldsymbol{a}_2), \boldsymbol{a}_2, \ldots, \boldsymbol{a}_{k-1}, (\boldsymbol{a}_{k-1}, \boldsymbol{a}_k), \boldsymbol{a}_k$, and the length of a path corresponds to its number of edges. For a path P, the length of P is denoted by $L(P)$. An edge $(\boldsymbol{a}, \boldsymbol{b})$ and a path from $\boldsymbol{a}$ to $\boldsymbol{b}$ can also be written as $\boldsymbol{a} \rightarrow \boldsymbol{b}$ and $\boldsymbol{a} \rightsquigarrow \boldsymbol{b}$, respectively.

Let us recall the theorem by Gu and Peng [10] relating to torus disjoint paths routing.

Theorem 1. [10] *In a (k,n)-torus, given two nodes $\boldsymbol{s}$ and $\boldsymbol{d}$, a set of $2n$ disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ ($1 \leq i \leq 2n$) of lengths at most $n\lfloor k/2 \rfloor + 1$ can be found in $O(n^3)$ time.*

3 Node-to-Set Disjoint Paths Routing Algorithm

We describe in this section an algorithm solving the node-to-set disjoint paths routing problem in a $TCC(k,n)$. Given a source node $\boldsymbol{s}$ and a set of destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}$ with $\boldsymbol{s}$, $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, and $\boldsymbol{d}_3$ all distinct, this algorithm generates mutually node-disjoint paths (except for $\boldsymbol{s}$) $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ ($1 \leq i \leq 3$).

The case $n = 1$ is special. Effectively, since each node of a $TCC(k,1)$ has only two neighbor nodes, three disjoint paths cannot be found. So, let us assume that $n \geq 2$. Also, algorithms that solve the container problem and the node-to-set disjoint paths routing problem in an n-dimensional torus are used. If $k = 2$, as each of these algorithms constructs $2n$ paths, then each node in the torus only has n neighbor nodes. Therefore, it is impossible to find $2n$ disjoint paths in this case as well. So, let us assume that $k \geq 3$. According to the relative positions of the nodes $\boldsymbol{s} = (\boldsymbol{c}(\boldsymbol{s}), p_{\boldsymbol{s}})$, $\boldsymbol{d}_1 = (\boldsymbol{c}(\boldsymbol{d}_1), p_{\boldsymbol{d}_1})$, $\boldsymbol{d}_2 = (\boldsymbol{c}(\boldsymbol{d}_2), p_{\boldsymbol{d}_2})$, and $\boldsymbol{d}_3 = (\boldsymbol{c}(\boldsymbol{d}_3), p_{\boldsymbol{d}_3})$, eight cases are distinguished.

3.1 Case 1: $D \subset C(\boldsymbol{s})$

We discuss in this subsection the case where $D \subset C(\boldsymbol{s})$.

Step 1 Without loss of generality, we can assume that $\boldsymbol{s}$, $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, and $\boldsymbol{d}_3$ are located in the cluster $C(\boldsymbol{s})$ in this order. In $C(\boldsymbol{s})$, construct two disjoint paths from $\boldsymbol{s}$ to $\boldsymbol{d}_1$ and $\boldsymbol{d}_3$.

Step 2 Select the external edge $\boldsymbol{s} \rightarrow \boldsymbol{n}_3(\boldsymbol{s})$.

Step 3 In $C(\boldsymbol{n}_3(\boldsymbol{s}))$, construct the shortest path $\boldsymbol{n}_3(\boldsymbol{s}) \rightsquigarrow (\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{s})), p_{\boldsymbol{d}_2})$.

Step 4 Select the external edge $(\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{s})), p_{\boldsymbol{d}_2}) \rightarrow \boldsymbol{s}' = \boldsymbol{n}_3(\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{s})), p_{\boldsymbol{d}_2})$.

Step 5 In $C(\boldsymbol{s}')$, construct the shortest path $\boldsymbol{s}' \rightsquigarrow (\boldsymbol{c}(\boldsymbol{s}'), p_{\boldsymbol{n}_3(\boldsymbol{s})})$.

Step 6 Select the external edge $(\boldsymbol{c}(\boldsymbol{s}'), p_{\boldsymbol{n}_3(\boldsymbol{s})}) \rightarrow (\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{d}_2)), p_{\boldsymbol{s}})$.

Step 7 In $C(\boldsymbol{n}_3(\boldsymbol{d}_2))$, construct the shortest path $(\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{d}_2)), p_{\boldsymbol{s}}) \rightsquigarrow \boldsymbol{n}_3(\boldsymbol{d}_2)$.

Step 8 Select the external edge $\boldsymbol{n}_3(\boldsymbol{d}_2) \rightarrow \boldsymbol{d}_2$. An illustration is given in Figure 2.

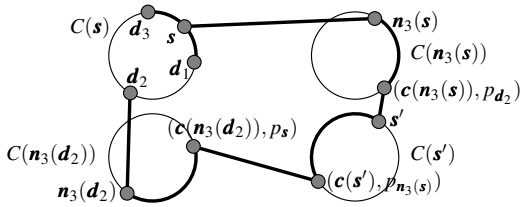


Figure 2: The three internally disjoint paths selected in Case 1

3.2 Case 2: $|D \cap C(\boldsymbol{s})| = 2$

In this subsection, we discuss the case where $|D \cap C(\boldsymbol{s})| = 2$.

Step 1 Without loss of generality, we can assume that $\boldsymbol{d}_1, \boldsymbol{d}_2 \in C(\boldsymbol{s})$. In $C(\boldsymbol{s})$, construct two disjoint paths from $\boldsymbol{s}$ to $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$.

Step 2 Apply the node-to-node disjoint paths routing algorithm in the (k,n)-torus where the source node is $\boldsymbol{c}(\boldsymbol{s})$ and the destination node is $\boldsymbol{c}(\boldsymbol{d}_3)$ to obtain a set of $2n$ disjoint paths P from $\boldsymbol{c}(\boldsymbol{s})$ to $\boldsymbol{c}(\boldsymbol{d}_3)$.

Step 3 Let p ($\in P$) be the path that includes the node $\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{s}))$. Convert p into the path $p' : \boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_3 (\in C(\boldsymbol{d}_3))$ in $TCC(k,n)$.

Step 4 In $C(\boldsymbol{d}_3)$, construct the shortest path $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$. An illustration is given in Figure 3.

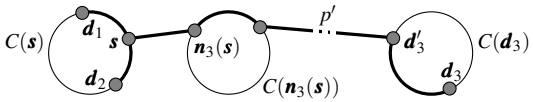


Figure 3: The three internally disjoint paths selected in Case 2

3.3 Case 3: $|D \cap C(\boldsymbol{s})| = 1$ and $\exists \boldsymbol{a} (\neq \boldsymbol{s})$ such that $|D \cap C(\boldsymbol{a})| = 2$

We now discuss the case where $|D \cap C(\boldsymbol{s})| = 1$ and $\exists \boldsymbol{a} (\neq \boldsymbol{s})$ such that $|D \cap C(\boldsymbol{a})| = 2$.

Step 1 Without loss of generality, we can assume that $\boldsymbol{d}_1 \in C(\boldsymbol{s})$. In $C(\boldsymbol{s})$, construct the shortest path from $\boldsymbol{s}$ to $\boldsymbol{d}_1$. We can also assume that the path includes $\boldsymbol{n}_1(\boldsymbol{s})$ but not $\boldsymbol{n}_2(\boldsymbol{s})$.

Step 2 Apply the node-to-node disjoint paths routing algorithm in the (k,n)-torus where the source node is $\boldsymbol{c}(\boldsymbol{s})$ and the destination node is $\boldsymbol{c}(\boldsymbol{a})$ to obtain a set of $2n$ disjoint paths P from $\boldsymbol{c}(\boldsymbol{s})$ to $\boldsymbol{c}(\boldsymbol{a})$.

Step 3 Let p_2 and p_3 ($\in P$) be the paths that include $\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{s}))$ and $\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{n}_2(\boldsymbol{s})))$, respectively. Convert p_2 and p_3 into the paths $p'_2 : \boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_2 (\in C(\boldsymbol{a}))$ and $p'_3 : \boldsymbol{n}_2(\boldsymbol{s}) \rightsquigarrow \boldsymbol{d}'_3 (\in C(\boldsymbol{a}))$ in $TCC(k,n)$, respectively.

Step 4 Select the internal edge $\boldsymbol{s} \rightarrow \boldsymbol{n}_2(\boldsymbol{s})$.

Step 5 In $C(\boldsymbol{a})$, construct the disjoint paths $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$ and $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$, or $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_3$ and $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_2$ depending on the relative positions of $\boldsymbol{d}'_2$, $\boldsymbol{d}_2$, $\boldsymbol{d}'_3$, and $\boldsymbol{d}_3$. An illustration is given in Figure 4.

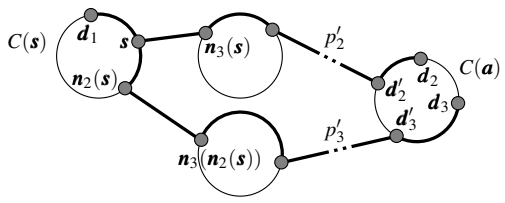


Figure 4: The three internally disjoint paths selected in Case 3

3.4 Case 4: $|D \cap C(\boldsymbol{s})| = 1$ and $\forall \boldsymbol{a} (\neq \boldsymbol{s})$ such that $|D \cap C(\boldsymbol{a})| \leq 1$

Now we discuss the case where $|D \cap C(\boldsymbol{s})| = 1$ and $\forall \boldsymbol{a} (\neq \boldsymbol{s})$ such that $|D \cap C(\boldsymbol{a})| \leq 1$.

Step 1 Without loss of generality, we can assume that $\boldsymbol{d}_1 \in C(\boldsymbol{s})$. If $D \cap C(\boldsymbol{n}_3(\boldsymbol{d}_1)) \neq \emptyset$, go to Step 7.

Step 2 Select a set of $2n$ clusters $\mathscr{C}$ so that the following conditions are satisfied.

1. $(\mathscr{C} \setminus \{C(\boldsymbol{d}_2), C(\boldsymbol{d}_3)\}) \subset N(C(\boldsymbol{s}))$.
2. If $C(\boldsymbol{n}_3(\boldsymbol{s})) \notin \{C(\boldsymbol{d}_2), C(\boldsymbol{d}_3)\}$, $C(\boldsymbol{n}_3(\boldsymbol{s})) \notin \mathscr{C}$.
3. $C(\boldsymbol{n}_3(\boldsymbol{d}_1)), C(\boldsymbol{d}_2), C(\boldsymbol{d}_3) \in \mathscr{C}$.

Step 3 In the (k,n)-torus, apply the node-to-set disjoint paths routing algorithm to obtain a set of $2n$ disjoint paths P where the source node is $\boldsymbol{c}(\boldsymbol{s})$ and the destination nodes are the nodes induced by the clusters in $\mathscr{C}$.

Step 4 Let p_2, p_3 ($\in P$) be the paths from $\boldsymbol{c}(\boldsymbol{s})$ to $\boldsymbol{c}(\boldsymbol{d}_2)$ and $\boldsymbol{c}(\boldsymbol{d}_3)$, respectively. Convert p_2, and p_3 into the paths in $TCC(k,n)$, $p'_2 : \boldsymbol{s}'_2 \rightsquigarrow \boldsymbol{d}'_2$, and $p'_3 : \boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}'_3$, respectively. We can assume without loss of generality that $\boldsymbol{s}'_2 = \boldsymbol{s}$.

Step 5 In $C(\boldsymbol{s})$, construct two disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$.

Step 6 Construct the shortest paths $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$ and $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ in $C(\boldsymbol{d}_2)$ and $C(\boldsymbol{d}_3)$, respectively. Terminate. An illustration is given in Figure 5a.

Step 7 We can assume without loss of generality that $\boldsymbol{d}_2 \in C(\boldsymbol{n}_3(\boldsymbol{d}_1))$. Select a node $\boldsymbol{b}$ such that $\boldsymbol{c}(\boldsymbol{b}) \neq \boldsymbol{c}(\boldsymbol{s}), \boldsymbol{c}(\boldsymbol{b}) \neq \boldsymbol{c}(\boldsymbol{d}_3)$, and there is an external edge $\boldsymbol{e}_1 \rightarrow \boldsymbol{e}_2$ between $C(\boldsymbol{d}_2)$ and $C(\boldsymbol{b})$ where $\boldsymbol{e}_1 \in C(\boldsymbol{d}_2)$ and $\boldsymbol{e}_2 \in C(\boldsymbol{b})$.

Step 8 Select a set of $2n$ clusters $\mathscr{C}$ so that the following conditions are satisfied.

1. $(\mathscr{C} \setminus \{C(\boldsymbol{b}), C(\boldsymbol{d}_3)\}) \subset N(C(\boldsymbol{s}))$.
2. If $C(\boldsymbol{n}_3(\boldsymbol{s})) \neq C(\boldsymbol{d}_3), C(\boldsymbol{n}_3(\boldsymbol{s})) \notin \mathscr{C}$.
3. $C(\boldsymbol{d}_2), C(\boldsymbol{b}), C(\boldsymbol{d}_3) \in \mathscr{C}$.

Step 9 In the (k,n)-torus, apply the node-to-set disjoint paths routing algorithm to obtain a set of $2n$ disjoint paths P where the source node is $\boldsymbol{c}(\boldsymbol{s})$ and the destination nodes are the nodes induced by the clusters in $\mathscr{C}$.

Step 10 Let p_2, p_3 ($\in P$) be the paths from $\boldsymbol{c}(\boldsymbol{s})$ to $\boldsymbol{c}(\boldsymbol{b})$ and $\boldsymbol{c}(\boldsymbol{d}_3)$, respectively. Convert p_2, and p_3 into the paths in $TCC(k,n)$, $p'_2 : \boldsymbol{s}'_2 \rightsquigarrow \boldsymbol{d}'_2 (\in C(\boldsymbol{b}))$,and $p'_3 : \boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}'_3$, respectively. We can assume that $\boldsymbol{s}'_2 = \boldsymbol{s}$ without loss of generality.

Step 11 In $C(\boldsymbol{s})$, construct the disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$.

Step 12 In $C(\boldsymbol{b})$, construct the shortest path $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{e}_2$.

Step 13 Select the edge $\boldsymbol{e}_2 \rightarrow \boldsymbol{e}_1$.

Step 14 In $C(\boldsymbol{d}_2)$, construct the shortest path $\boldsymbol{e}_1 \rightsquigarrow \boldsymbol{d}_2$.

Step 15 In $C(\boldsymbol{d}_3)$, construct the shortest path $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$. An illustration is given in Figure 5b.

3.5 Case 5: $\exists \boldsymbol{a} (\neq \boldsymbol{s})$ such that $|D \cap C(\boldsymbol{a})| = 3$

We now discuss the case where $\exists \boldsymbol{a} (\neq \boldsymbol{s})$ such that $|D \cap C(\boldsymbol{a})| = 3$.

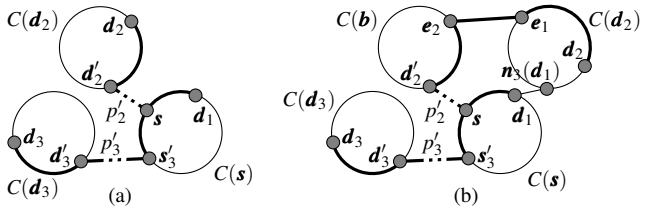


Figure 5: The three internally disjoint paths selected in Case 4

Step 1 In the (k,n)-torus, apply the node-to-node disjoint paths routing algorithm to obtain a set of $2n$ disjoint paths P where the source and destination nodes are $\boldsymbol{c}(\boldsymbol{s})$ and $\boldsymbol{c}(\boldsymbol{a})$, respectively.

Step 2 Assume that p_1 ($\in P$) is the path that includes $\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{s}))$. Convert p_1 into the path $p'_1 : \boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_1$ in $TCC(k,n)$. Without loss of generality, we can assume that $\boldsymbol{d}_1$ is the nearest node from $\boldsymbol{d}'_1$ in $C(\boldsymbol{a})$.

Step 3 Let p_2 and p_3 ($\in P$) be the paths that include $\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{d}_2))$ and $\boldsymbol{c}(\boldsymbol{n}_3(\boldsymbol{d}_3))$, respectively. Convert p_2 and p_3 into the paths $p'_2 : \boldsymbol{s}'_2 (\in C(\boldsymbol{s})) \rightsquigarrow \boldsymbol{d}_2 (\in C(\boldsymbol{a}))$ and $p'_3 : \boldsymbol{s}'_3 (\in C(\boldsymbol{s})) \rightsquigarrow \boldsymbol{d}_3 (\in C(\boldsymbol{a}))$, respectively.

Step 4 In $C(\boldsymbol{s})$, construct two disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$. An illustration is given in Figure 6.

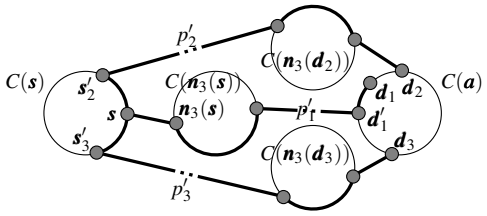


Figure 6: The three internally disjoint paths selected in Case 5

3.6 Case 6: $D \cap C(\boldsymbol{s}) = \emptyset$ and $\exists \boldsymbol{a} (\neq \boldsymbol{s})$ such that $|D \cap C(\boldsymbol{a})| = 2$

We discuss in this subsection the case where $D \cap C(\boldsymbol{s}) = \emptyset$ and $\exists \boldsymbol{a} (\neq \boldsymbol{s})$ such that $|D \cap C(\boldsymbol{a})| = 2$.

Step 1 We can assume without loss of generality that $\boldsymbol{d}_1, \boldsymbol{d}_2 \in C(\boldsymbol{a})$. Select a node $\boldsymbol{b}$ such that $\boldsymbol{c}(\boldsymbol{b}) \neq \boldsymbol{c}(\boldsymbol{s}), \boldsymbol{c}(\boldsymbol{b}) \neq \boldsymbol{c}(\boldsymbol{d}_3)$, and there is an external edge $\boldsymbol{e}_1 \rightarrow \boldsymbol{e}_2$ between $C(\boldsymbol{b})$ and $C(\boldsymbol{a})$ where $\boldsymbol{e}_1 \in C(\boldsymbol{b})$ and $\boldsymbol{e}_2 \in C(\boldsymbol{a})$.

Step 2 Select a set of $2n$ clusters $\mathscr{C}$ so that the following conditions are satisfied.

1. $(\mathscr{C} \setminus \{C(\boldsymbol{a}), C(\boldsymbol{b}), C(\boldsymbol{d}_3)\}) \subset N(C(\boldsymbol{s}))$.
2. If $C(\boldsymbol{n}_3(\boldsymbol{s})) \notin \{C(\boldsymbol{a}), C(\boldsymbol{b}), C(\boldsymbol{d}_3)\}$, $C(\boldsymbol{n}_3(\boldsymbol{s})) \notin \mathscr{C}$.
3. $C(\boldsymbol{a}), C(\boldsymbol{b}), C(\boldsymbol{d}_3) \in \mathscr{C}$.

Step 3 In (k,n)-torus, apply the node-to-set disjoint paths routing algorithm to obtain a set of $2n$ disjoint paths P where the source node is $\boldsymbol{c}(\boldsymbol{s})$ and the destination nodes are induced by the clusters in $\mathscr{C}$.

Step 4 Let p_1, p_2, p_3 ($\in P$) be the paths from $\boldsymbol{c}(\boldsymbol{s})$ to $\boldsymbol{c}(\boldsymbol{a})$, $\boldsymbol{c}(\boldsymbol{b})$, and $\boldsymbol{c}(\boldsymbol{d}_3)$, respectively. Convert p_1, p_2, and p_3 into the paths in $TCC(k,n)$, $p'_1 : \boldsymbol{s}'_1 \rightsquigarrow \boldsymbol{d}'_1$, $p'_2 : \boldsymbol{s}'_2 \rightsquigarrow \boldsymbol{d}'_2$, and $p'_3 : \boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}'_3$, respectively.

Step 5 Without loss of generality, we can assume that $\boldsymbol{s}'_1 = \boldsymbol{s}$, $\boldsymbol{s}'_2 \neq \boldsymbol{s}$, and $\boldsymbol{s}'_3 \neq \boldsymbol{s}$. In $C(\boldsymbol{s})$, construct two disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$.

Step 6 Without loss of generality, we can assume that $\boldsymbol{d}'_1 \in C(\boldsymbol{a})$, $\boldsymbol{d}'_2 \in C(\boldsymbol{b})$, and $\boldsymbol{d}'_3 \in C(\boldsymbol{d}_3)$. In $C(\boldsymbol{d}_3)$, construct the shortest path $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$.

Step 7 In $C(\boldsymbol{b})$, construct the shortest path $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{e}_1$. Select the edge $\boldsymbol{e}_1 \rightarrow \boldsymbol{e}_2$.

Step 8 In $C(\boldsymbol{a})$, construct the disjoint paths $\boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{e}_2 \rightsquigarrow \boldsymbol{d}_2$, or $\boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{d}_2$ and $\boldsymbol{e}_2 \rightsquigarrow \boldsymbol{d}_1$ depending on the relative positions of $\boldsymbol{d}'_1$, $\boldsymbol{d}_1$, $\boldsymbol{e}_2$, and $\boldsymbol{d}_2$. An illustration is given in Figure 7.

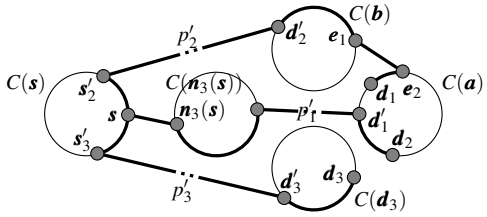


Figure 7: The three internally disjoint paths selected in Case 6

3.7 Case 7: $D \cap C(\boldsymbol{s}) = \emptyset$, $\forall \boldsymbol{a}, |D \cap C(\boldsymbol{a})| \leq 1$, and $D \subset N(C(\boldsymbol{s})) \setminus \{C(\boldsymbol{n}_3(\boldsymbol{s}))\}$

Now we discuss the case where $D \cap C(\boldsymbol{s}) = \emptyset$, $\forall \boldsymbol{a}, |D \cap C(\boldsymbol{a})| \leq 1$, and $D \subset N(C(\boldsymbol{s})) \setminus \{C(\boldsymbol{n}_3(\boldsymbol{s}))\}$.

Step 1 Let $\boldsymbol{s}'_2 (\in C(\boldsymbol{s})) \rightarrow \boldsymbol{d}'_2 (\in C(\boldsymbol{d}_2))$ be the external edge between $C(\boldsymbol{s})$ and $C(\boldsymbol{d}_2)$, and $\boldsymbol{s}'_3 (\in C(\boldsymbol{s})) \rightarrow \boldsymbol{d}'_3 (\in C(\boldsymbol{d}_3))$ be the external edge between $C(\boldsymbol{s})$ and $C(\boldsymbol{d}_3)$. Select a node $\boldsymbol{b}$ such that $C(\boldsymbol{b}) \in N(C(\boldsymbol{d}_1))$ and $\boldsymbol{c}(\boldsymbol{b}) \neq \boldsymbol{c}(\boldsymbol{s})$. Let $\boldsymbol{e}_1 (\in C(\boldsymbol{d}_1)) \rightarrow \boldsymbol{e}_2 (\in C(\boldsymbol{b}))$ be the external edge between $C(\boldsymbol{d}_1)$ and $C(\boldsymbol{b})$.

Step 2 Let $\mathscr{C} = (N(C(\boldsymbol{s})) \setminus \{C(\boldsymbol{n}_3(\boldsymbol{s}))\}) \cup \{C(\boldsymbol{b})\}$.

Step 3 In the (k,n)-torus, apply the node-to-set disjoint paths routing algorithm to obtain a set of $2n$ disjoint paths P where the source node is $\boldsymbol{c}(\boldsymbol{s})$ and the destination nodes are induced by the clusters in $\mathscr{C}$.

Step 4 Let p ($\in P$) be the path from $\boldsymbol{c}(\boldsymbol{s})$ to $\boldsymbol{c}(\boldsymbol{b})$. Convert p into the path in $TCC(k,n)$, $p' : \boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_1$.

Step 5 In $C(\boldsymbol{b})$, construct the shortest path $\boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{e}_2$.

Step 6 Select the external edge $\boldsymbol{e}_2 \rightarrow \boldsymbol{e}_1$.

Step 7 In $C(\boldsymbol{d}_1)$, construct the shortest path $\boldsymbol{e}_2 \rightsquigarrow \boldsymbol{d}_1$.

Step 8 In $C(\boldsymbol{s})$, construct two disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$.

Step 9 Select the external edge $\boldsymbol{s}'_2 \rightarrow \boldsymbol{d}'_2$.

Step 10 In $C(\boldsymbol{d}_2)$, construct the shortest path $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$.

Step 11 Select the external edge $\boldsymbol{s}'_3 \rightarrow \boldsymbol{d}'_3$.

Step 12 In $C(\boldsymbol{d}_3)$, construct the shortest path $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$. An illustration is given in Figure 8.

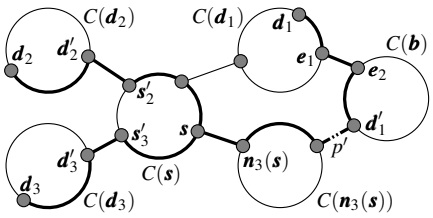


Figure 8: The three internally disjoint paths selected in Case 7

3.8 Case 8: $D \cap C(\boldsymbol{s}) = \emptyset$, $\forall \boldsymbol{a}, |D \cap C(\boldsymbol{a})| \leq 1$, and $D \notin N(C(\boldsymbol{s})) \setminus \{C(\boldsymbol{n}_3(\boldsymbol{s}))\}$

We now discuss the case where $D \cap C(\boldsymbol{s}) = \emptyset, \forall \boldsymbol{a}, |D \cap C(\boldsymbol{a})| \leq 1$, and $D \notin N(C(\boldsymbol{s})) \setminus \{C(\boldsymbol{n}_3(\boldsymbol{s}))\}$.

Step 1 Select a set of $2n$ clusters $\mathscr{C}$ so that the following conditions are satisfied.

1. $(\mathscr{C} \setminus \{C(\boldsymbol{d}_1), C(\boldsymbol{d}_2), C(\boldsymbol{d}_3)\}) \subset N(C(\boldsymbol{s}))$.
2. If $C(\boldsymbol{n}_3(\boldsymbol{s})) \notin \{C(\boldsymbol{d}_1), C(\boldsymbol{d}_2), C(\boldsymbol{d}_3)\}$, $C(\boldsymbol{n}_3(\boldsymbol{s})) \notin \mathscr{C}$.
3. $C(\boldsymbol{d}_1), C(\boldsymbol{d}_2), C(\boldsymbol{d}_3) \in \mathscr{C}$.

Step 2 In the (k,n)-torus, apply the node-to-set disjoint paths routing algorithm to obtain a set of $2n$ disjoint paths P where the source node is $\boldsymbol{c}(\boldsymbol{s})$ and the destination nodes are induced by the clusters in $\mathscr{C}$.

Step 3 Let p_1, p_2, p_3 ($\in P$) be the paths from $\boldsymbol{c}(\boldsymbol{s})$ to $\boldsymbol{c}(\boldsymbol{d}_1)$, $\boldsymbol{c}(\boldsymbol{d}_2)$, and $\boldsymbol{c}(\boldsymbol{d}_3)$, respectively. Convert p_1, p_2, and p_3 into the paths in $TCC(k,n)$, $p'_1 : \boldsymbol{s}'_1 \rightsquigarrow \boldsymbol{d}'_1$, $p'_2 : \boldsymbol{s}'_2 \rightsquigarrow \boldsymbol{d}'_2$, and $p'_3 : \boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}'_3$, respectively.

Step 4 We can assume that $\boldsymbol{s}'_1 = \boldsymbol{s}$ without loss of generality. In $C(\boldsymbol{s})$, construct disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$.

Step 5 In $C(\boldsymbol{d}_1)$, construct the shortest path $\boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{d}_1$.

Step 6 In $C(\boldsymbol{d}_2)$, construct the shortest path $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$.

Step 7 In $C(\boldsymbol{d}_3)$, construct the shortest path $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$. An illustration is given in Figure 9.

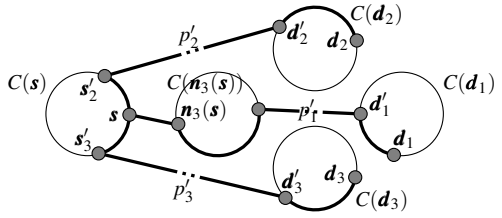


Figure 9: The three internally disjoint paths selected in Case 8

4 Performance Evaluation

In this section, we prove the correctness of our algorithm, estimate the maximum length of the disjoint paths obtained and the algorithm time complexity. Note that we use the notation κ to represent $\lfloor k/2 \rfloor$.

Lemma 1. *For a source node $\boldsymbol{s}$ and a set of three destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}$ in a $TCC(k,n)$ where $n \geq 2$ and $k \geq 3$, the algorithm of Section 3 generates three disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ ($1 \leq i \leq 3$) of lengths at most $3n+4$ in $O(n)$ time in Case 1.*

Proof. Since the two paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_3$ traverse the cycle of $C(\boldsymbol{s})$ in opposite directions, they are disjoint except for $\boldsymbol{s}$. In addition, they do not include $\boldsymbol{d}_2$ because of the relative positions of $\boldsymbol{s}$, $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, and $\boldsymbol{d}_3$. The path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_2$ is outside of $C(\boldsymbol{s})$ except for the terminal nodes $\boldsymbol{s}$ and $\boldsymbol{d}_2$. Therefore, the three paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ ($1 \leq i \leq 3$) are disjoint except for $\boldsymbol{s}$.

The lengths of the two paths constructed in Step 1 are at most $2n-3$. The length of the path constructed in Steps 2 to 8 is at most $1+n+1+n+1+n+1 = 3n+4$. Therefore, the maximum path length in Case 1 is $3n+4$.

It takes $O(n)$ time to construct two disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_3$ in $C(\boldsymbol{s})$ in Step 1. Each selection of the edges in Steps 2, 4, 6 and 8 takes $O(1)$ time. Each construction of the shortest paths in Steps 3, 5 and 7 takes $O(n)$ time. Hence, the total time complexity of Case 1 is $O(n)$. □

Lemma 2. *For a source node $\boldsymbol{s}$ and a set of three destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}$ in a $TCC(k,n)$ where $n \geq 2$ and $k \geq 3$, the algorithm of Section 3 generates three disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ ($1 \leq i \leq 3$) of lengths at most $\kappa n^2 + (\kappa+1)n + 1$ in $O(n^3 + kn^2)$ time in Case 2.*

Proof. Since the two paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_2$ traverse the cycle of $C(\boldsymbol{s})$ in opposite directions, they are disjoint except for $\boldsymbol{s}$. The path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_3$ is outside of $C(\boldsymbol{s})$ except for the terminal node $\boldsymbol{s}$. So, the three paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ ($1 \leq i \leq 3$) are disjoint except for $\boldsymbol{s}$. The lengths of two paths constructed in Step 1 are at most $2n-2$.

From Theorem 1, the path p obtained in Step 3 has length $L(p)$ of at most $\kappa n + 1$. Therefore, the converted path p' has length $L(p')$ of at most $L(p) + (L(p)-1) \times n = \kappa n^2 + \kappa n + 1$. The length of the sub path $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ constructed in Step 4 is at most n. Hence, the length of the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ is at most $(\kappa n^2 + \kappa n + 1) + n = \kappa n^2 + (\kappa+1)n + 1$. If $k \geq 3$, $\kappa n^2 + (\kappa+1)n + 1 \geq 2n-2$. Therefore, the maximum path length in Case 2 is $\kappa n^2 + (\kappa+1)n + 1$.

It takes $O(n)$ time to construct two disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_2$ in $C(\boldsymbol{s})$ in Step 1. From Theorem 1, the time complexity of Step 2 is $O(n^3)$. In Step 3, p can be found in $O(n)$ time by checking the second nodes of $2n$ disjoint paths. Since $L(p)$ is $O(kn)$, conversion of p into p' in Step 3 takes $O(kn^2)$ time. Construction of the shortest paths in Step 4 takes $O(n)$ time. Hence, the total time complexity of Case 2 is $O(n^3 + kn^2)$. □

Lemma 3. *For a source node $\boldsymbol{s}$ and a set of three destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}$ in a $TCC(k,n)$ where $n \geq 2$ and $k \geq 3$, the algorithm of Section 3 generates three disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ ($1 \leq i \leq 3$) of lengths at most $\kappa n^2 + (\kappa+2)n - 1$ in $O(n^3 + kn^2)$ time in Case 3.*

Proof. Since the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and the edge $\boldsymbol{s} \rightarrow \boldsymbol{n}_2(\boldsymbol{s})$ traverse the cycle of $C(\boldsymbol{s})$ in opposite directions, they are disjoint except for $\boldsymbol{s}$. The sub paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_2$ and $\boldsymbol{n}_2(\boldsymbol{s}) \rightsquigarrow \boldsymbol{d}'_3$ visit different clusters except for $C(\boldsymbol{s})$ and $C(\boldsymbol{a})$. In $C(\boldsymbol{a})$, the sub paths

$\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$ and $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ (or $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_3$ and $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_2$) are disjoint. Hence, the paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_3$ are disjoint except for $\boldsymbol{s}$. They are also disjoint with the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ except for $\boldsymbol{s}$.

The length of the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ constructed in Step 1 is at most n. Similarly to the proof for Case 2, $L(p'_2)$ and $L(p'_3)$ are both at most $\kappa n^2 + \kappa n + 1$. Here we assume that the sub paths $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ and $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$ are taken in Step 5 without loss of generality. Since the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$ includes the sub path $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$ of length at most $2n-3$, its length is at most $(\kappa n^2 + \kappa n + 1) + (2n - 3) = \kappa n^2 + (\kappa + 2)n - 2$. Since the path $\boldsymbol{s} \rightarrow \boldsymbol{n}_2(\boldsymbol{s}) \rightsquigarrow \boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ includes the edge $\boldsymbol{s} \rightarrow \boldsymbol{n}_2(\boldsymbol{s})$ and the sub path $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ of length at most $2n-3$, its length is at most $1 + (\kappa n^2 + \kappa n + 1) + (2n - 3) = \kappa n^2 + (\kappa + 2)n - 1$. Therefore, the maximum path length in Case 3 is $\kappa n^2 + (\kappa + 2)n - 1$.

It takes $O(n)$ time to construct the shortest path in $C(\boldsymbol{s})$ in Step 1. From Theorem 1, the time complexity of Step 2 is $O(n^3)$. In Step 3, p_2 and p_3 can be found in $O(n)$ time by checking the second nodes of $2n$ disjoint paths. Since $L(p_2)$ and $L(p_3)$ are both $O(kn)$, their conversion into p'_2 and p'_3 in Step 3 takes $O(kn^2)$ time. Selection of the edge in Step 4 takes $O(1)$ time. Construction of the two paths in Step 5 takes $O(n)$ time. Hence, the total time complexity of Case 3 is $O(n^3 + kn^2)$. □

Lemma 4. *For a source node $\boldsymbol{s}$ and a set of three destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}$ in a TCC(k,n) where $n \geq 2$ and $k \geq 3$, the algorithm of Section 3 generates three disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \leq i \leq 3)$ of lengths at most $\kappa n^2 + (\kappa + 2)n + 2$ if $n = 2$ and $\kappa n^2 + (\kappa + 3)n - 1$ otherwise in $O(n^3 + kn^2)$ time in Case 4.*

Proof. First, we consider the sub case where $D \cap C(\boldsymbol{n}_3(\boldsymbol{d}_1)) \neq \emptyset$. Since the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and the sub path $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$ traverse the cycle of $C(s)$ in opposite directions, they are disjoint except for $\boldsymbol{s}$. The path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_2$ and the sub path $\boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}_3$ visit different clusters except for $C(\boldsymbol{s})$. Hence, the paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_3$ are disjoint except for $\boldsymbol{s}$. They are also disjoint with the remaining path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ except for $\boldsymbol{s}$.

From Theorem 1, the maximum length of the disjoint paths obtained in Step 3 is $\kappa n + 1$. Then the maximum length of the two disjoint paths p'_2 and p'_3 obtained in Step 4 is $\kappa n^2 + \kappa n + 1$. The lengths of the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and the sub paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$ constructed in Step 5 are both at most $2n-2$. In Step 6, the lengths of the sub paths $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$ in $C(\boldsymbol{d}_2)$ and $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ in $C(\boldsymbol{d}_3)$ are both at most n. Hence, the maximum path length in this sub case is attained by the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$, and it is $(2n-2) + (\kappa n^2 + \kappa n + 1) + n = \kappa n^2 + (\kappa + 3)n - 1$.

It takes $O(1)$ time to check if $D \cap C(\boldsymbol{n}_3(\boldsymbol{d}_1)) \neq \emptyset$ or not in Step 1. It takes $O(n)$ time to select $2n$ clusters that satisfy the conditions in Step 2. From Theorem 1, the time complexity of Step 3 is $O(n^3)$. In Step 4, p_2 and p_3 can be found in $O(n)$ time by checking the final nodes of $2n$ disjoint paths. Since $L(p_2)$ and $L(p_3)$ are both $O(kn)$, their conversion into p'_2 and p'_3 takes $O(kn^2)$ time in Step 4. Construction of two disjoint paths in $C(\boldsymbol{s})$ in Step 5 takes $O(n)$ time. Construction of two shortest paths in $C(\boldsymbol{d}_2)$ and $C(\boldsymbol{d}_3)$ in Step 6 takes $O(n)$ time. Hence, the time complexity of this sub case is $O(n^3 + kn^2)$.

Next, we consider the sub case where $D \cap C(\boldsymbol{n}_3(\boldsymbol{d}_1)) = \emptyset$. Since the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and the sub path $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$ traverse the cycle of $C(\boldsymbol{s})$ in opposite directions, they are disjoint except for $\boldsymbol{s}$. In the (k,n)-torus, $\boldsymbol{c}(\boldsymbol{d}_2)$ is included in $\mathscr{C}$. Therefore, the set of $2n$ disjoint paths P obtained by the node-to-set disjoint paths routing algorithm includes the path $\boldsymbol{c}(\boldsymbol{s}) \rightsquigarrow \boldsymbol{c}(\boldsymbol{d}_2)$ though it is discarded later. Hence, the cluster $C(\boldsymbol{d}_2)$ cannot be visited by the sub paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_2$ and $\boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}'_3$. Also, the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_2$ and the sub path $\boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}_3$ visit different clusters except for $C(\boldsymbol{s})$. So, the paths $\boldsymbol{s} \rightsquigarrow d_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_3$ are disjoint except for $\boldsymbol{s}$. They are also disjoint with the remaining path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ except for $\boldsymbol{s}$.

Again, from Theorem 1, the maximum length of the disjoint paths obtained in Step 9 is $\kappa n + 1$. Then the maximum length of the two disjoint paths p'_2 and p'_3 obtained in Step 10 is $\kappa n^2 + \kappa n + 1$. The lengths of the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ and the sub path $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$ constructed in Step 11 are both at most $2n-2$. In Step 12, the length of the sub path $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{e}_2$ in $C(\boldsymbol{s})$ is at most n. The shortest path $\boldsymbol{e}_1 \rightsquigarrow \boldsymbol{d}_2$ in $C(\boldsymbol{d}_2)$ constructed in Step 14 has length of at most n. The length of the shortest path $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ in $C(\boldsymbol{d}_3)$ in Step 15 is at most n. Hence, the lengths of the paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$, $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{e}_2 \rightarrow \boldsymbol{e}_1 \rightsquigarrow \boldsymbol{d}_2$, and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ are at most $2n-2$, $(\kappa n^2 + \kappa n + 1) + n + 1 + n = \kappa n^2 + (\kappa + 2)n + 2$, and $(2n-2) + (\kappa n^2 + \kappa n + 1) + n = \kappa n^2 + (\kappa + 3)n - 1$, respectively. Therefore, the maximum path length in this sub case is $\kappa n^2 + (\kappa + 2)n + 2$ if $n = 2$ and $\kappa n^2 + (\kappa + 3)n - 1$ otherwise.

It takes $O(1)$ time to check if $D \cap C(\boldsymbol{n}_3(\boldsymbol{d}_1)) \neq \emptyset$ or not in Step 1. It takes $O(n)$ time to select the node $\boldsymbol{b}$ and the edge $\boldsymbol{e}_1 \rightarrow \boldsymbol{e}_2$ that satisfy the conditions in Step 7. It takes $O(n)$ time to select $2n$ clusters that satisfy the conditions in Step 8. From Theorem 1, the time complexity of Step 9 is $O(n^3)$. In Step 10, p_2 and p_3 can be found in $O(n)$ time by checking the final nodes of $2n$ disjoint paths. Since $L(p_2)$ and $L(p_3)$ are both $O(kn)$, their conversion into p'_2 and p'_3 takes $O(kn^2)$ time in Step 10. Construction of two disjoint paths in $C(\boldsymbol{s})$ in Step 11 takes $O(n)$ time. Construction of the shortest path in $C(\boldsymbol{b})$ in Step 12 takes $O(n)$ time. Selection of the edge in Step 13 takes $O(1)$ time. Construction of the shortest path in $C(\boldsymbol{d}_2)$ in Step 14 takes $O(n)$ time. Construction of the shortest path in $C(\boldsymbol{d}_3)$ in Step 15 takes $O(n)$ time. Hence, the time complexity of this sub case is $O(n^3 + kn^2)$.

Consequently, the path length in this case is at most $\kappa n^2 + (\kappa + 2)n + 2$ if $n = 2$ and $\kappa n^2 + (\kappa + 3)n - 1$ otherwise. Also, Case 4 is $O(n^3 + kn^2)$ total time. □

The remaining cases are proved similarly. So, we skip the proofs of Lemmas 5, 6, 7 and 8 below.

Lemma 5. *For a source node $\boldsymbol{s}$ and a set of three destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}$ in a TCC(k,n) where $n \geq 2$ and $k \geq 3$, the algorithm of Section 3 generates three disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \leq i \leq 3)$ of lengths at most $\kappa n^2 + (\kappa + 2)n - 1$ in $O(n^3 + kn^2)$ time in Case 5.*

Proof. Since the sub paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$ traverse the cycle of $C(\boldsymbol{s})$ in opposite directions, they are disjoint except for $\boldsymbol{s}$. Because the sub paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_1$, $\boldsymbol{s}'_2 \rightsquigarrow \boldsymbol{d}_2$, and $\boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}_3$ are based

on the disjoint paths generated by the node-to-node disjoint paths routing algorithm in the (k,n)-torus, they visit different clusters except for $C(\boldsymbol{s})$ and $C(\boldsymbol{a})$. In $C(\boldsymbol{a})$, the sub path $\boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{d}_1$ does not include $\boldsymbol{d}_2$ nor $\boldsymbol{d}_3$ since $\boldsymbol{d}_1$ is assumed to be the nearest destination node from $\boldsymbol{d}'_1$. Hence, the paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \leq i \leq 3)$ are disjoint except for $\boldsymbol{s}$.

From Theorem 1, the maximum length of the disjoint paths obtained in Step 1 is $\kappa n+1$. The length of the path p'_1 obtained in Step 2 is at most $\kappa n^2+\kappa n+1$. Similarly, the maximum length of the paths p'_2 and p'_3 obtained in Step 3 is $\kappa n^2+\kappa n+1$. The lengths of the two disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$ in Step 4 are both at most $2n-2$. Therefore, the lengths of the paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{d}_1$, $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2 \rightsquigarrow \boldsymbol{d}_2$, and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}_3$ are at most $(\kappa n^2+\kappa n+1)+(n-1) = \kappa n^2+(\kappa+1)n$, $(2n-2)+(\kappa n^2+\kappa n+1) = \kappa n^2+(\kappa+2)n-1$, and $(2n-2)+(\kappa n^2+\kappa n+1) = \kappa n^2+(\kappa+2)n-1$, respectively. Hence, in this case, the maximum path length is $\kappa n^2+(\kappa+2)n-1$.

From Theorem 1, the time complexity of Step 1 is $O(n^3)$. In Step 2, p_1 can be found in $O(n)$ time by checking the second nodes of $2n$ paths. Since $L(p_1)$ is $O(kn)$, its conversion into p'_1 takes $O(kn^2)$ time in Step 2. The nearest destination node $\boldsymbol{d}_1$ from $\boldsymbol{d}'_1$ can be found in $O(1)$ time in Step 2. In Step 3, p_2 and p_3 can be found in $O(n)$ time by checking the semi-final nodes of $2n$ disjoint paths. Their conversion into p'_2 and p'_3 in Step 3 takes $O(kn^2)$ time since $L(p_2)$ and $L(p_3)$ are both $O(kn)$. Construction of two disjoint paths in $C(\boldsymbol{s})$ in Step 4 takes $O(n)$ time. Consequently, the total time complexity of Case 5 is $O(n^3+kn^2)$. □

Lemma 6. *For a source node $\boldsymbol{s}$ and a set of three destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}$ in a TCC(k,n) where $n \geq 2$ and $k \geq 3$, the algorithm of Section 3 generates three disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \leq i \leq 3)$ of lengths at most $\kappa n^2+(\kappa+5)n-4$ in $O(n^3+kn^2)$ time in Case 6.*

Proof. Since the sub paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$ traverse the cycle of $C(\boldsymbol{s})$ in opposite directions, they are disjoint except for $\boldsymbol{s}$. Because the sub paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_1$, $\boldsymbol{s}'_2 \rightsquigarrow \boldsymbol{d}_2$, and $\boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}_3$ are based on the disjoint paths generated by the node-to-set disjoint paths routing algorithm in the (k,n)-torus, they visit different clusters except for $C(\boldsymbol{s})$. The sub paths $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{e}_1$ and $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ are disjoint with other paths since they are inside the clusters $C(\boldsymbol{b})$ and $C(\boldsymbol{d}_3)$, respectively. There is only one external edge $\boldsymbol{e}_1 \rightarrow \boldsymbol{e}_2$ between $C(\boldsymbol{b})$ and $C(\boldsymbol{a})$. Therefore, $\boldsymbol{e}_2 \neq \boldsymbol{d}'_1$. The sub paths $\boldsymbol{e}_2 \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{d}_2$ (or $\boldsymbol{e}_2 \rightsquigarrow \boldsymbol{d}_2$ and $\boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{d}_1$) are disjoint. Hence, the paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \leq i \leq 3)$ are disjoint except for $\boldsymbol{s}$.

From Theorem 1, the maximum length of the disjoint paths obtained in Step 3 is $\kappa n+1$. The lengths of the three sub paths $p'_1 : \boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_1$, $p'_2 : \boldsymbol{s}'_2 \rightsquigarrow \boldsymbol{d}'_2$, and $p'_3 : \boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}'_3$ are both at most $\kappa n^2+\kappa n+1$. The lengths of the disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$ constructed in Step 5 are both at most $2n-2$. The shortest path $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ constructed in Step 6 has length of at most n. The shortest path $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{e}_1$ constructed in Step 7 has length of at most n. The maximum lengths of the two disjoint paths, say $\boldsymbol{e}_2 \rightsquigarrow \boldsymbol{d}_1$ and $\boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{d}_2$ constructed in Step 8 are both $2n-3$. Hence, the lengths of the paths $\boldsymbol{s} \rightsquigarrow s'_2 \rightsquigarrow \boldsymbol{e}_2 \rightsquigarrow \boldsymbol{d}_1$, $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{d}_2$, and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ are at most $\kappa n^2+(\kappa+2)n-2$, $\kappa n^2+(\kappa+5)n-4$,and $\kappa n^2+(\kappa+3)n-1$, respectively. Consequently, the maximum path length in Case 6 is $\kappa n^2+(\kappa+5)n-4$.

It takes $O(n)$ time to select the node $\boldsymbol{b}$ and the edge $\boldsymbol{e}_1 \rightarrow \boldsymbol{e}_2$ that satisfy the conditions in Step 1. It takes $O(n)$ time to select $2n$ clusters that satisfy the conditions in Step 2. From Theorem 1, the time complexity of Step 3 is $O(n^3)$. In Step 4, p_1, p_2 and p_3 can be found in $O(n)$ time by checking the final nodes of $2n$ disjoint paths. Since $L(p_1)$, $L(p_2)$, and $L(p_3)$ are all $O(kn)$, their conversion into p'_1, p'_2, and p'_3 in Step 4 takes $O(kn^2)$ time. Construction of two disjoint paths in $C(\boldsymbol{s})$ in Step 5 takes $O(n)$ time. Construction of the shortest path in $C(\boldsymbol{d}_3)$ in Step 6 takes $O(n)$ time. Construction of the shortest path and selection of the edge in Step 7 take $O(n)$ and $O(1)$, respectively. Construction of two disjoint paths in $C(\boldsymbol{a})$ in Step 8 takes $O(n)$. Consequently, the total time complexity of Case 6 is $O(n^3+kn^2)$. □

Lemma 7. *For a source node $\boldsymbol{s}$ and a set of three destination nodes $D = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \boldsymbol{d}_3\}$ in a TCC(k,n) where $n \geq 2$ and $k \geq 3$, the algorithm of Section 3 generates three disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_i$ $(1 \leq i \leq 3)$ of lengths at most $\kappa n^2+(\kappa+2)n+2$ in $O(n^3+kn^2)$ time in Case 7.*

Proof. Since the sub paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$ traverse the cycle of $C(\boldsymbol{s})$ in opposite directions, they are disjoint except for $\boldsymbol{s}$. In the (k,n)-torus, $\boldsymbol{c}(\boldsymbol{d}_1)$ is included in $\mathscr{C}$. Therefore, the set of $2n$ disjoint paths P obtained by the node-to-set disjoint paths routing algorithm includes the path $\boldsymbol{c}(\boldsymbol{s}) \rightsquigarrow \boldsymbol{c}(\boldsymbol{d}_1)$ though it is discarded later. Hence, the cluster $C(\boldsymbol{d}_1)$ cannot be visited by the sub path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_1$. The sub path $\boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{d}_1$ visits $C(\boldsymbol{b})$ and $C(\boldsymbol{d}_1)$, which are not visited by other paths. Also, the path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_2$ and the sub path $\boldsymbol{s}'_3 \rightsquigarrow \boldsymbol{d}_3$ visit different clusters except for $C(\boldsymbol{s})$. Hence, the paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_3$ are disjoint except for $\boldsymbol{s}$. They are also disjoint with the remaining path $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}_1$ except for $\boldsymbol{s}$.

The maximum length of the disjoint paths obtained in Step 3 is $\kappa n+1$ from Theorem 1. The path length $L(p')$ of $p' : \boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_1$ obtained in Step 4 is at most $\kappa n^2+\kappa n+1$. The lengths of the four shortest paths $\boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{e}_2$ in $C(\boldsymbol{b})$ in Step 5, $\boldsymbol{e}_2 \rightsquigarrow \boldsymbol{d}_1$ in $C(\boldsymbol{d}_1)$ in Step 7, $\boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$ in $C(\boldsymbol{d}_2)$ in Step 10, and $\boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ in $C(\boldsymbol{d}_3)$ in Step 12 are all at most n. The maximum length of the disjoint paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2$ and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3$ in $C(\boldsymbol{s})$ constructed in Step 8 is $2n-2$. Therefore, the lengths of the paths $\boldsymbol{s} \rightsquigarrow \boldsymbol{d}'_1 \rightsquigarrow \boldsymbol{e}_2 \rightarrow \boldsymbol{e}_1 \rightsquigarrow \boldsymbol{d}_1$, $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_2 \rightarrow \boldsymbol{d}'_2 \rightsquigarrow \boldsymbol{d}_2$, and $\boldsymbol{s} \rightsquigarrow \boldsymbol{s}'_3 \rightarrow \boldsymbol{d}'_3 \rightsquigarrow \boldsymbol{d}_3$ are at most $(\kappa n^2+\kappa n+1)+n+1+n = \kappa n^2+(\kappa+2)n+2$, $(2n-2)+1+n = 3n-1$, and $(2n-2)+1+n = 3n-1$, respectively. Consequently, the maximum path length in Case 7 is $\kappa n^2+(\kappa+2)n+2$.

It takes $O(n)$ time to select the node $\boldsymbol{b}$ and the edge $\boldsymbol{e}_1 \rightarrow \boldsymbol{e}_2$ that satisfy the conditions in Step 1. $\mathscr{C}$ can be constructed in $O(n)$ in Step 2. From Theorem 1, the time complexity of Step 3 is $O(n^3)$. In Step 4, p can be found in $O(n)$ time by checking the final nodes of $2n$ disjoint paths. Conversion of p into p' takes $O(kn^2)$ time since $L(p)$ is $O(kn)$. Construction of the shortest paths in Steps 5, 7, 10 and 12 takes $O(n)$ time. Construction of two disjoint paths in $C(\boldsymbol{s})$ in Step 8 takes $O(n)$ time. Selection

of edges in Steps 6, 9 and 11 takes $O(1)$ time. Consequently, the total time complexity of Case 7 is $O(n^3+kn^2)$. □

Lemma 8. *For a source node $\boldsymbol{s}$ and a set of three destination nodes $D=\{\boldsymbol{d}_1,\boldsymbol{d}_2,\boldsymbol{d}_3\}$ in a TCC(k,n) where $n\geq 2$ and $k\geq 3$, the algorithm of Section 3 generates three disjoint paths $\boldsymbol{s}\rightsquigarrow\boldsymbol{d}_i$ $(1\leq i\leq 3)$ of lengths at most $\kappa n^2+(\kappa+3)n-1$ in $O(n^3+kn^2)$ time in Case 8.*

Proof. Since the sub paths $\boldsymbol{s}\rightsquigarrow\boldsymbol{s}'_2$ and $\boldsymbol{s}\rightsquigarrow\boldsymbol{s}'_3$ traverse the cycle of $C(\boldsymbol{s})$ in opposite directions, they are disjoint except for $\boldsymbol{s}$. Because the sub paths $\boldsymbol{s}\rightsquigarrow\boldsymbol{d}'_1$, $\boldsymbol{s}'_2\rightsquigarrow\boldsymbol{d}'_2$, and $\boldsymbol{s}'_3\rightsquigarrow\boldsymbol{d}'_3$ are based on the disjoint paths generated by the node-to-set disjoint paths routing algorithm in the (k,n)-torus, they visit different clusters except for $C(\boldsymbol{s})$. The sub paths $\boldsymbol{d}'_1\rightsquigarrow\boldsymbol{d}_1$, $\boldsymbol{d}'_2\rightsquigarrow\boldsymbol{d}_2$, and $\boldsymbol{d}'_3\rightsquigarrow\boldsymbol{d}_3$ are disjoint with other paths since they are inside the clusters $C(\boldsymbol{d}_1)$, $C(\boldsymbol{d}_2)$, and $C(\boldsymbol{d}_3)$, respectively. Hence, the paths $\boldsymbol{s}\rightsquigarrow\boldsymbol{d}_i$ $(1\leq i\leq 3)$ are disjoint except for $\boldsymbol{s}$.

The maximum length of the disjoint paths obtained in Step 2 is $\kappa n+1$ from Theorem 1. Then, the lengths of the sub paths $p'_1:\boldsymbol{s}'_1\rightsquigarrow\boldsymbol{d}'_1$, $p'_2:\boldsymbol{s}'_2\rightsquigarrow\boldsymbol{d}'_2$, and $p'_3:\boldsymbol{s}'_3\rightsquigarrow\boldsymbol{d}'_3$ that are obtained in Step 3 are at most $\kappa n^2+\kappa n+1$. The maximum lengths of the sub paths $\boldsymbol{s}\rightsquigarrow\boldsymbol{s}'_2$ and $\boldsymbol{s}\rightsquigarrow\boldsymbol{s}'_3$ in $C(\boldsymbol{s})$ constructed in Step 4 are both $2n-2$. The shortest paths $\boldsymbol{d}'_1\rightsquigarrow\boldsymbol{d}_1$ in $C(\boldsymbol{d}_1)$ constructed in Step 5, $\boldsymbol{d}'_2\rightsquigarrow\boldsymbol{d}_2$ in $C(\boldsymbol{d}_2)$ constructed in Step 6, and $\boldsymbol{d}'_3\rightsquigarrow\boldsymbol{d}_3$ in $C(\boldsymbol{d}_3)$ in Step 7 have lengths of at most n. Hence, the lengths of the paths $\boldsymbol{s}\rightsquigarrow\boldsymbol{d}'_1\rightsquigarrow\boldsymbol{d}_1$, $\boldsymbol{s}\rightsquigarrow\boldsymbol{s}'_2\rightsquigarrow\boldsymbol{d}'_2\rightsquigarrow\boldsymbol{d}_2$, and $\boldsymbol{s}\rightsquigarrow\boldsymbol{s}'_3\rightsquigarrow\boldsymbol{d}'_3\rightsquigarrow\boldsymbol{d}_3$ are at most $(\kappa n^2+\kappa n+1)+n=\kappa n^2+(\kappa+1)n+1$, $(2n-2)+(\kappa n^2+\kappa n+1)+n=\kappa n^2+(\kappa+3)n-1$, and $(2n-2)+(\kappa n^2+\kappa n+1)+n=\kappa n^2+(\kappa+3)n-1$, respectively. Consequently, the maximum path length in Case 8 is $\kappa n^2+(\kappa+3)n-1$.

It takes $O(n)$ time to select $2n$ clusters that satisfy the conditions in Step 1. From Theorem 1, the time complexity of Step 2 is $O(n^3)$. In Step 3, p_1, p_2 and p_3 can be found in $O(n)$ time by checking the final nodes of $2n$ disjoint paths. Since $L(p_1)$, $L(p_2)$, and $L(p_3)$ are all $O(kn)$, their conversion into p'_1, p'_2, and p'_3 in Step 3 takes $O(kn^2)$ time. Construction of two disjoint paths in $C(\boldsymbol{s})$ in Step 4 takes $O(n)$ time. Construction of the shortest paths in Steps 5, 6 and 7 takes $O(n)$ time. Consequently, the total time complexity of Case 8 is $O(n^3+kn^2)$. □

Finally, we can recapitulate this discussion into the following theorem.

Theorem 2. *For a source node $\boldsymbol{s}$ and a set of three destination nodes $D=\{\boldsymbol{d}_1,\boldsymbol{d}_2,\boldsymbol{d}_3\}$ in a TCC(k,n) where $n\geq 2$ and $k\geq 3$, the algorithm of Section 3 generates three disjoint paths $\boldsymbol{s}\rightsquigarrow\boldsymbol{d}_i$ $(1\leq i\leq 3)$ of lengths at most $\kappa n^2+(\kappa+5)n-4$ in $O(n^3+kn^2)$ time.*

Proof. This can be directly deduced from Lemmas 1 to 8. □

5 Conclusion

Disjoint paths routing is critical for reliability and performance of parallel systems. In this paper, we have proposed a new algorithm that solves the node-to-set disjoint paths routing problem in a TCC. For any source node $\boldsymbol{s}$ and a set of three destination nodes $D=\{\boldsymbol{d}_1,\boldsymbol{d}_2,\boldsymbol{d}_3\}$ in a TCC(k,n) where $n\geq 2$ and $k\geq 3$, the algorithm constructs three disjoint paths $\boldsymbol{s}\rightsquigarrow\boldsymbol{d}_1$, $\boldsymbol{s}\rightsquigarrow\boldsymbol{d}_2$, and $\boldsymbol{s}\rightsquigarrow\boldsymbol{d}_3$. The maximum length of the constructed paths and the time complexity of the algorithm are proved to be $\kappa n^2+(\kappa+5)n-4$ where $\kappa=\lfloor k/2\rfloor$ and $O(n^3+kn^2)$, respectively. Regarding previous works, it would be interesting to solve the set-to-set and pairwise disjoint paths routing problems in a TCC network. Also, cluster fault-tolerance is an important topic.

Acknowledgements

This study was partly supported by the Research Foundation for the Electrotechnology of Chubu and by a Grant-in-Aid for Scientific Research (C) of the Japan Society for the Promotion of Science under Grant No. 25330079.

References

[1] A. Bossard, K. Kaneko, and S. Peng, "A New Node-to-Set Disjoint-Path Algorithm in Perfect Hierarchical Hypercubes," *Comp. J.*, 54(8):1372–1381, 2011.

[2] A. Bossard and K. Kaneko, "The Set-to-Set Disjoint-Path Problem in Perfect Hierarchical Hypercubes," *Comp. J.*, 55(6):769–775, 2012.

[3] A. Bossard and K. Kaneko, "Node-to-Set Disjoint-Path Routing in Hierarchical Cubic Networks," *Comp. J.*, 55(12):1440–1446, 2012.

[4] A. Bossard and K. Kaneko, "Torus-Connected Cycles: an Implementation-Friendly Topology for Interconnection Networks of Massively Parallel Systems," *Proc. Int. Conf. Alg. Arch. Par. Proc.*, Vietri sul Mare, Italy, pp. 11–21, Dec. 18–20, 2013.

[5] A. Bossard and K. Kaneko, "Set-to-Set Disjoint Paths Routing in Hierarchical Cubic Networks," *Comp. J.*, 57(2):332–337, 2014.

[6] A. Bossard and K. Kaneko, "The Container Problem in a Torus-Connected Cycles Network," *Proc. Int. Conf. Comp. Sci.*, Cairns, Australia, pp. 2182–2191, June 10–12, 2014.

[7] A. Bossard and K. Kaneko, "Node-to-Set Disjoint Paths Routing in a Torus-Connected Cycles Network," *Proc. Int. Conf. Comp. Applications in Industry and Eng.*, New Orleans, LA, USA, pp. 135–140, Oct. 13–15, 2014.

[8] K. Ghose and K.R. Desai, "The HCN: a Versatile Interconnection Network Based on Cubes," *Proc. 1989 ACM/IEEE Conf. Supercomp.*, Reno, NV, USA, pp. 426–435, Nov. 12–17, 1989.

[9] Q.-P. Gu and S. Peng, "An Efficient Algorithm for Set-to-Set Node-Disjoint Paths Problem in Hypercubes," *Proc. Int. Conf. Par. Distr. Sys.*, Tokyo, Japan, pp. 98–105, June 3–6, 1996.

[10] Q.-P. Gu and S. Peng, "Fault Tolerant Routing in Toroidal Networks," *IEICE Trans. Inf. Syst.*, E79-D(8):1153–1159, 1996.

[11] Q.-P. Gu and S. Peng, "Set-to-Set Fault Tolerant Routing in Star Graphs," *IEICE Trans. Inf. Syst.*, E79-D(4):282–289, 1996.

[12] Q.-P. Gu and S. Peng, "Node-to-Set Disjoint Paths Problem in Star Graphs," *Inf. Proc. Let.*, 62(4):201–207, 1997.

[13] K. Kaneko, "An Algorithm for Node-to-Set Disjoint Paths Problem in Burnt Pancake Graphs," *IEICE Trans. Inf. Syst.*, E86-D(12):2588–2594, 2003.

[14] K. Kaneko and N. Sawada, "An Algorithm for Node-to-Node Disjoint Paths Problem in Burnt Pancake Graphs," *IEICE Trans. Inf. Syst.*, E90-D(1):306–313, 2007.

[15] K. Kaneko and Y. Suzuki, "Node-to-Set Disjoint Paths Problem in Pancake Graphs," *IEICE Trans. Inf. Syst.*, E86-D(9):1628–1633, 2003.

[16] Y. Li, S. Peng, and W. Chu, "Efficient Collective Communications in Dual-Cube," *J. Supercomp.*, 28(1):71–90, 2004.

[17] Y. Li, S. Peng, and W. Chu, "Metacube - a Versatile Family of Interconnection Networks for Extremely Large-Scale Supercomputers," *J. Supercomp.*, 53(2):329–351, 2010.

[18] Q. M. Malluhi and M. A. Bayoumi, "The Hierarchical Hypercube: a New Interconnection Topology for Massively Parallel Systems," *IEEE Trans. Par. Dist. Syst.*, 5(1):17–30, 1994.

[19] J. Moroo, M. Yamada, and T. Kato, "Operation System for the K Computer," *Fujitsu Sci. Tech. J.*, 48(3):295–301, 2012.

[20] S. Peng and K. Kaneko, "Set-to-Set Disjoint Paths Routing in Pancake Graphs," *Proc. Int. Conf. Par. Distr. Comp. Syst.*, Dallas, TX, USA, pp. 253–258, Nov. 13–15, 2006.

[21] M. O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance," *J. Assoc. Comp. Mach.*, 36(2):335–348, 1989.

[22] Y. Saad and M. H. Schultz, "Topological Properties of Hypercubes," *IEEE Trans. Comp.*, 37(7):867–872, 1988.

[23] Y. Suzuki and K. Kaneko, "An Algorithm for Node-Disjoint Paths in Pancake Graphs," *IEICE Trans. Inf. Syst.*, E86-D(3):610–615, 2003.

[24] T. Takabatake, K. Kaneko, and H. Ito, "Generalized Hierarchical Completely-Connected Network," *Proc. Int. Symp. Par. Arch., Alg. Net.*, Perth/Fremantle, Australia, pp. 68–73, June 23–25, 1999.

Antoine Bossard is an Assistant Professor at the Advanced Institute of Industrial Technology, Tokyo Metropolitan University. His research is focused on graph theory, interconnection networks and dependable systems. He received the B.E. and M.E. degrees from Université de Caen Basse-Normandie, France in 2005 and 2007, respectively, and the Ph.D. degree from Tokyo University of Agriculture and Technology in 2011. He is a member of ACM and ISCA.

Keiichi Kaneko is a Professor at Tokyo University of Agriculture and Technology. His main research areas are dependable systems, interconnection networks, functional programming, parallel and distributed computation, partial evaluation and educational systems. He received the B.E., M.E. and Ph.D. degrees from the University of Tokyo in 1985, 1987 and 1994, respectively. He is a member of IEEE, ACM, IEICE, IPSJ and JSSST.

CMAC-Based Computational Model of Affects (CCMA) from Self-Organizing Feature Mapping Weights for Classification of Emotion Using EEG Signals

Hamwira Yaacob*, Wahab Abdul*
International Islamic University Malaysia, Kuala Lumpur, MALAYSIA

Norhaslinda Kamaruddin
MARA University of Technology (UiTM), Selangor, MALAYSIA

Abstract

Emotion is postulated to be generated at the brain. To capture the brain activities during emotional processing, several neuro-imaging techniques have been adopted, including *electroencephalogram* (EEG). In the existing studies, different techniques have been employed to extract features from EEG signals for emotion classification. However, existing feature extraction techniques do not consider spatial and temporal neural-dynamics of emotion. Furthermore, the non-linearity of EEG and self-adaptive of neural activations are disregard. Therefore, the classification accuracy of any feature extraction technique is inconsistent when applied with different classifiers. Hence, in this study, a new feature extraction technique that inculcates the qualities of EEG signal and the behavior of neural activations is proposed based on Cerebellar Model Articulation Controller (CMAC) model. The accuracy of classifying calm, fear, happiness and sadness emotional states using Evolving Fuzzy Neural Network (EFuNN) classifiers are reported based on subject-dependent and subject-independent validations. The classification performance of using features from power spectral density (PSD), kernel density estimation (KDE) and mel-frequency ceptral coefficients (MFCC) are also compared and reported. It is observed that the proposed technique is able to yield accuracy of above 50% to above 90% for subject-dependent classification. For subject-independent approach, the highest accuracy is barely 40%. The results suggest that this approach is comparable as a feature extraction technique for classifying emotions.

Key Words: Affective computing, encephalogram (EEG), cerebellar model of articulation controller (CMAC), evolving fuzzy neural network (EFuNN), valence, arousal.

1 Introduction

A circumplex model of affect [34] defines emotion based on two affective dimensions, namely valence and arousal. It can be represented as a two dimensional space, in which the horizontal axis is the valence and the vertical axis is for arousal. The level of valence defines the positive and negative of an emotional state. On the other hand, the arousal axis determines the intensity of emotion from inactive to aroused states. The model can be adapted to identify emotion from different emotional expressions including automatic nervous systems (ANS) responses, behaviors (i.e., facial expressions, voice and speech, body language and posture as well as text) and the brain states [5].

Evidences based on over a decade of studies have suggested that the generations of emotion are also reflected at the brain. One of the earliest observations reported the change in personality of a construction worker, Phineas P. Gage, after recovering from surgery to remove an iron bar that passed through his head at a work incident in 1848 [14]. During the surgery, small parts of his brain were removed. Despite fully recovering physically, it was observed that he had become irreverent, impatient, quick to anger and unreliable. From a different observation, changes in emotional behaviors were also noticed in a number of cats that also experienced brain lesions [2]. More of the history and future directions of the studies concerning with the underlying neural correlates of emotions are discussed in [9].

With the emergence of more sophisticated non-invasive neuro-imaging techniques, studies of the brain no longer require an invasive technique that performs physically cutting open the skull and implanting electrodes on the brain. Among different non-invasive techniques, electroencephalogram (EEG) has become a prominent technique to capture brain activities since it is less costly compared to the others. Although EEG resolves poorly in capturing spatial information, relatively it has the highest temporal resolution. It captures brain activation up to milliseconds, which is comparable with the temporal dynamic of emotion [8]. Therefore, EEG has been widely employed for emotion recognition.

Recognizing emotion is seen as a pattern recognition problem. That is, a supervised learning method is implemented to classify emotion from features that are extracted using different techniques. Many techniques have been adapted to extract features from EEG signals including fractal dimension analysis [22, 40], power spectral density [3, 13, 21, 38], wavelet transform [27], statistical features [6, 24] and fusion of brain signal features and peripheral features. Such techniques do not consider the spatial and temporal dynamics of emotion

* Kulliyyah of ICT. Email: {hyaacob, abdulwahab}@iium.edu.my.
† Faculty of Computer & Mathematical Sciences, 40400 Shah Alam. Email: norhaslinda@tmsk.uitm.edu.my.

generation in the brain. Furthermore, the non-linearity of EEG and self-adaptive of neural activations are not implemented. Therefore, the classification accuracy of any feature extraction technique is dependent upon selected classifiers.

Hence, in this paper, a new feature extraction technique based on a neural inspired computational model, namely Cerebellar Model Articulation Controller (CMAC), is introduced. CMAC is considered because it is inculcated with the capabilities to perform self-organization feature mapping (SOFM) for non-linear problems. These qualities are also identified in brain signals that are captured by EEG based on the theory of chaotic system [41] as explained by the concepts of non-linear dynamic systems [17]. Thus, the proposed feature extraction technique is called CMAC-based Computational Model of Affect (CCMA-Type I).

Furthermore, Evolving Fuzzy Neural Network (EFuNN) is employed to perform emotion classification based on subject-dependent and subject-independent validations. A brief literature review is presented in Section 2. Section 3 elaborates the structure and functionalities of the proposed model. The methodology is presented in Section 4. The remainder of the paper discusses the results and conclusion in Section 5 and Section 6, respectively.

2 Previous Works

Most of emotional recognition studies that use EEG signals as the input involve several steps, including signal acquisition, pre-processing, feature extraction and classification. EEG signals are acquired by presenting the corresponding stimuli to the participants while EEG electrodes are attached at the scalp for recording the brain activation activities. Different numbers of electrodes are used in different studies. Stimuli presented to the participants during the signal acquisitions to elicit emotions include facial images, video clips, music pieces, speeches and many more.

In the pre-processing step, noise in the brain signals is removed either through visual inspection or existing computational techniques. At this point, the corresponding frequency bands are selected among delta (δ: 1–3 Hz), theta (θ: 4–7 Hz), alpha (α: 8–13 Hz), beta (β: 14–30 Hz), and gamma (γ: 31–50 Hz). No specific frequency bands are applied for the emotion recognition problem. For example, all bands are included in the studies conducted by [21, 23, 26, 40]. In different studies, only a few bands are selected. From the literatures cover in this study, in most studies used alpha and beta bands are commonly included.

To extract features from EEG signals, many techniques have been adapted, such as fractal dimension analysis [22, 40], power spectral density [3, 13, 21, 39], wavelet transform [27], statistical features [6, 24] and fusion of brain signal features and peripheral features. Extracted features are trained to produce model for the corresponding problems. However, in some studies, feature selection step [21, 23, 39] is performed to reduce dimensionality of the problem, hence reduce the training complexity.

Classification of emotional states is performed by machine learning classifiers. Output of the classifiers is determined based on the emotion classification techniques derived from psychological understandings. Emotions are perceived to be either *discreet* or *continuous*. According to the theories of basic emotions, emotions are discreet, such that each emotional state is different than other emotions as perceived from psychological and physiological manifestations. Different researchers have come up with different lists of basic emotions, as discussed in [11, 29, 32, 44].

Other researchers believe that an emotional state can be determined by the position of an intersecting point of several continuous dimensions in a space. Several dimensional models have been proposed in [4, 25, 32, 34, 36] to define the affective space. However, most commonly a two-dimensional affective space of valence and arousal is commonly adapted. Summary of the previous works in emotion classification using EEG is presented in Table 1.

3 CMAC-based Computational Model of Affect (CCMA) from Self-Organizing Feature Map

Cerebellar Model Articulation Controller (CMAC) is inspired by the functions of human cerebellum. It was initially proposed to model non-linear functions of robotic controllers [1]. However, over time, the original architecture and processes of CMAC have evolved. Pseudo-self-evolving CMAC (PSECMAC) [43] was proposed to overcome some limitations of the original CMAC such as poor memory utilization and generalization accuracy dilemma. In short, PSECMAC is just an implementation of a single-layered quantitization function CMAC.

Figure 1(a) illustrates the learning map of single-layered CMAC with two dimensional inputs, X_1 and X_2. Each input dimension is quantized into 5 discrete quantitization levels. Represented in a 2-dimensional matrix, 121 weighted memory cells are formed as the learning map. The aim of CMAC learning is to extract an optimal learning map that produces minimum errors between the observed output and the estimated output. Estimated outputs are calculated from the aggregation of memory contents associated with the activated input vectors.

Similar to other supervised learning approaches, the error between the calculated output and the desired output, Y_n, of the corresponding cell is calculated to improve the predictive performance of a model. In CMAC, the learning process involves the weights updates in the associative memory.

In contrast to the conventional CMAC network implementation, instead of only one winner cell determined for a particular instance, the adaptive learning of feature map in CCMA involves winner cells that are determined based on the coordinates of all EEG electrodes that are projected onto the feature map. In addition to that, each winner cell activates several neighboring cells based on an arbitrary radius, R, which delineates the region of activations. The activated region is defined as a square matrix which edge is R cells away from the winner cell along the horizontal and the vertical axis in both directions. Moreover, the activation factor intensity decreases from the center of the activated region towards the edge.

Features are extracted using the proposed CMAC-based

Table 1: Summary of previous works on emotion classification using EEG

Sources	Stimuli	Emotion model	EEG features	Feature Extraction	Classifiers	Accuracy
[45]	Music	Neutral to arousal	Theta, Alpha, Beta	• Normalized EEG power	SVM	About 80%
[6]	Facial images	Discrete		• EEG statistical features	MLP	49.17% to 90%
[13]	Music	Like - dislike	Beta, Gamma	• Time-Frequency Distributions • PSD, • the Zhao-Atlas-Marks method, • the Hilbert-Huang spectrum	kNN SVM	
[3]	Database for Emotion Analysis using Physiological Signal (DEAP)	Valence-arousal	4.0 to 45.0 Hz	• Statistical characteristics, • PSD (Power Spectral Density) • HOC (High Order Crossings)	kNN	69.59% to 70.1%
[39]	Video	Arousal - valence	Theta, Alpha, Beta, Gamma	• Logarithms of the PSD	SVM	50.5% to 62.1%
[26]	Facial images	Discrete	Delta, Theta, Alpha, Beta, Gamma	• Common Spatial Patterns (CSP)	Bayesian; Linear SVM	72.8% to 97.9%
[24]	Academic emotions induced through modified Wisconsin Card Sorting Task (WCST), Berg's	Discrete	8Hz-30Hz	• EEG statistical features	SVM; MLP; kNN	40.72% to 54.09%
[47]	Images	Valence – arousal	Theta, Alpha, Beta, Gamma	• FFT with a Hamming window	ANOVA	About 90%
[40]	Music & sound	Valence-arousal	Delta, Theta, Alpha, Beta, Gamma	• Fractal dimension (FD)	SVM	70 % to 100 %
[33]	Facial images	Valence – arousal	Alpha	• Kernel Density Estimation (KDE)		60.74% to 71.84%
[31]	Images	Valence - arousal	Alpha, Beta	• High-order crossing • Cross-correlation	SVM	62.58% to 94.40%
[22]	Songs and other audio	Valence – arousal	2 to 42 Hz	• Fractal dimension	SVM	84.9%
[21]	Soundtrack	valence-arousal	Delta, Theta, Alpha, Beta, Gamma	• PSD	MLP; SVM	82.29%±3 .06%
[30]	Facial image	Discrete	Alpha , Beta	• Higher Order Crossings (HOCs) EEG statistical features	QDA; k-NN; MD; SVM	77.66 % to 85.17%
[19]	Visual and aural stimulus	Discrete		• EEG statistical features	RVM; MLP; Decision Tree; SVM; Bayesian	Above 86%
[23]	Images	Valence - arousal	Delta, Theta, Alpha, Beta, Gamma	• EEG statistical features	Bayesian	Average 76%
[20]	Facial images			• Common Spatial Patterns (CSP)	SVM	About 93%
[16]	Music		Theta, Alpha	• Kernel Density Estimation (KDE) • Gaussian Mixture Model (GMM)	Bayesian; MLP; One-Rule; Random Tree; RBF	Around 90%
[27]	Movie clips	Discrete emotions	Alpha	• Wavelet	FCM	
[7]	Images	Arousal	4-45Hz	• EEG statistical features	Bayesian; FDA	Around 53% to 72%

method as discussed in Section 2. The CMAC learning map is a 15-by-15 matrix which is derived based on the coordinates of 8 EEG channels as specified in EEGLab Toolbox [10], as shown in Figure 1(b.)

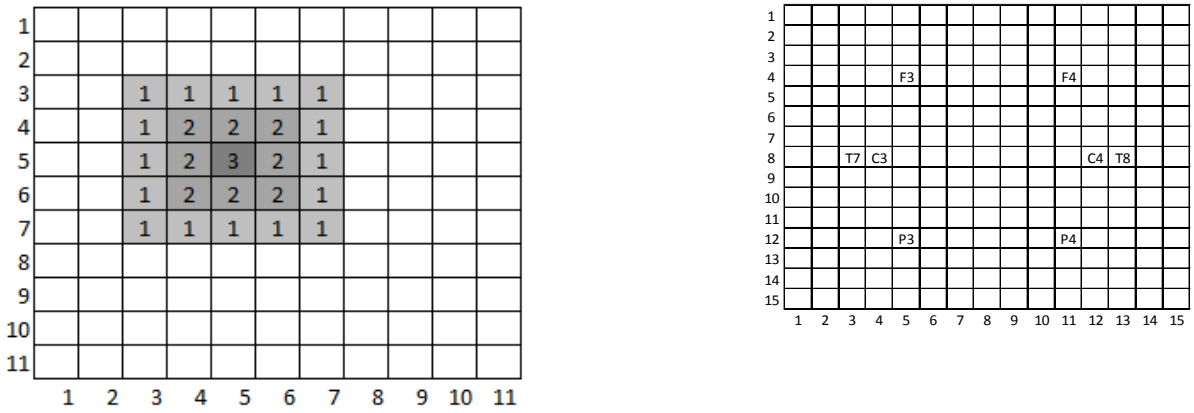
Initially, weights of the learning map are set to 0. As shown in the flowchart of adaptive learning in Figure 2, at each *training cycle*, the weights of the activated region, which are derived from each of the winner cells and the corresponding neighboring cells, are selected and updated. This is based on the error between calculated output, $y_{x,i}$, of the corresponding activated region and the desired output, $y'_{x,i}$.

The calculated output of the CMAC network for one channel corresponds to the ***k***-th activated neighbor at ***i***-th cycle and is calculated as Equation 1:

$$y_{x,i} = f(x) = \frac{\sum_{k=1}^{K} a_k(x) w_i}{\sum_{k=1}^{K} a_k(x)} \quad (1)$$

Where

x is the winner cell,
K is the total number of activated neighboring cells,
$a_k(x)$ is the activation factor at k-th neighboring cell of x,
w_i is the feature map weights at i -th cycle.



(a) (b)

Figure 1: (a) Single-layered CMAC learning map of two-dimensional input. As the input of respective dimensions, X_1 and X_2 determine the winner cell (as shaded in gray), (b) Projection of EEG channels on CMAC learning map

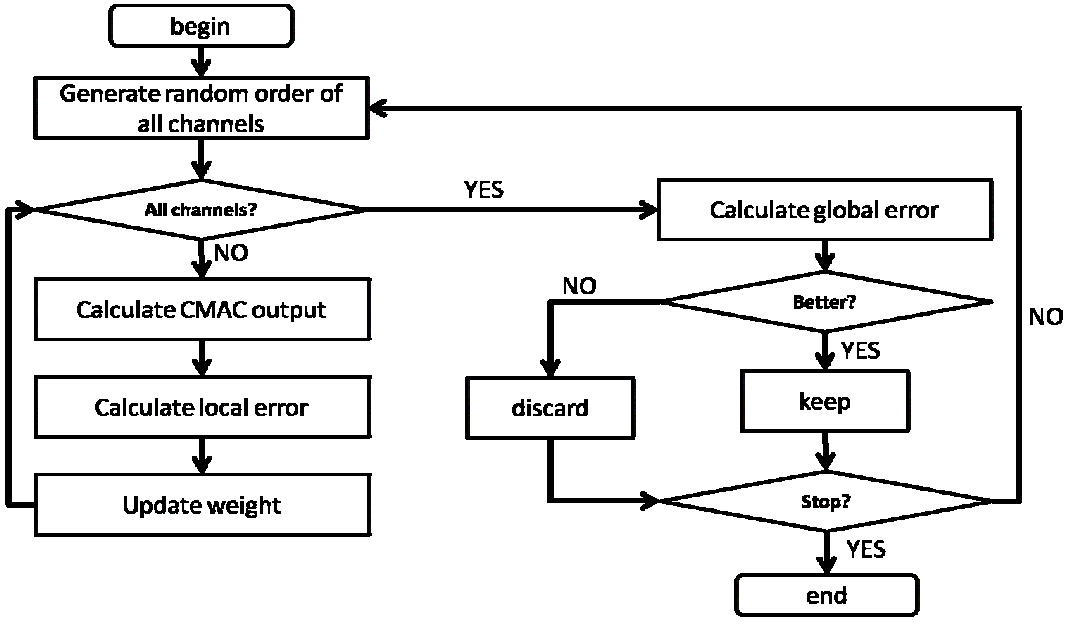


Figure 2: Adaptive learning of feature map for one instance

Thus, the local error, $\mathcal{L}_{k,i}$, is defined as:

$$\mathcal{L}_{x,i} = y'_{x,i} - y_{x,i} \quad (2)$$

where

$y'_{x,i}$ is the desired output based the power spectral density of each EEG signal at channel x for i -th cycle,

$y_{x,i}$ is the calculated output the corresponding input, x , at i -th cycle.

The weight update is described in Equation 1.

$$w_{k,i+1} = w_{k,i} + \alpha \mathcal{L}_{x,i} \frac{a_k(x)}{\sum_{k=1}^{K} a_k(x)} \quad (3)$$

Where

$w_{k,i+1}$ is the feature map weights correspond to the k -th activated neighbor at $(i+1)$-th cycle

$w_{k,i}$ is the feature map weights correspond to the k -th activated neighbor at i -th cycle

$\mathcal{L}_{x,i}$ is the local error corresponds to the k -th activated neighbor at i -th cycle,

$a_k(x)$ is the activation factor at *k*-th neighboring cell of x ,

K is the total number of activated neighboring cells,

α is an arbitrary learning rate constant.

After the weights of all winner cells and the corresponding neighboring cells of one instance have been updated, the *global error*, $\mathcal{G}_i$, is calculated based on the mean of *local errors*, $\overline{\mathcal{L}_{x,i}}$. An arbitrary threshold, Θ , is used to define the final by fulfilling the following condition:

$$w_x^* = \begin{cases} w_{i+1}, & \mathcal{G}_{i+1} < \mathcal{G}_i \\ w_i, & \mathcal{G}_{i+1} \geq \mathcal{G}_i \end{cases} \quad (6)$$

As a result, a feature set of the corresponding sample is extracted based on the region of activations which are defined by the corresponding activation radius, *R*. Next, an arbitrary supervised learning algorithm is implemented to perform classification for profiling emotions from the extracted feature vector. Based on the dimensional view of emotions, two supervised learning classifiers are employed representing the valence and arousal dimensions.

4 Methodology

By adapting the affective computing approach, this study implements several steps included to perform emotion classification, as depicted in Figure 3. The steps are data collection, signal pre-processing, feature extraction and classification.

4.1 Data Collection

EEG signals are recorded using the BIMEC from Brainmarker BV. It consists of eight channels which are placed on the participants' scalp based on the international 10-20 EEG electrode positioning system [28], namely: C3, C4, F3, F4, P3,

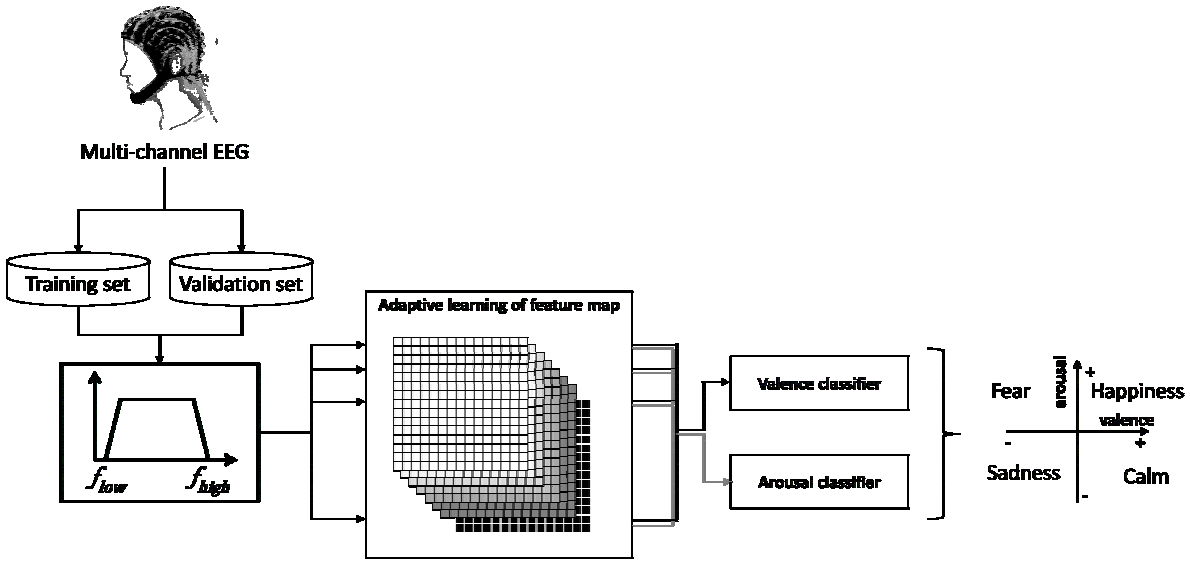


Figure 3: Methodology

P4, T7 and T8. As shown in Figure 4, the corresponding electrodes that are placed on the scalp are connected to the EEG electrode box which captures electrical discharge during the stimulated brain activities. The electrical discharges are then amplified through the BIMEC amplifier. The signals are recorded and stored at the signal analysis machine. Moreover, the stimuli presentation screen is placed in front the participants. In addition, a digital camera is also used to capture the behavior of participants.

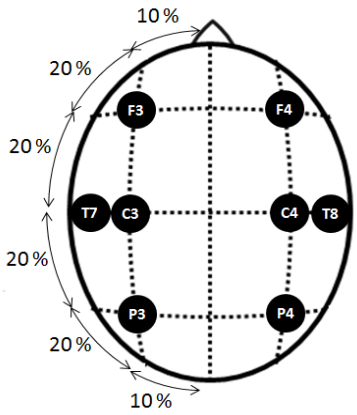


Figure 4: Placements of 8 electrodes based on 10-20 EEG electrodes positioning standard system

In this study, EEG signals are recorded from 11 normal and healthy children of both genders aged between 4 to 6 years old with written consent from the parents or guardian. It is started by recording 1 minute of eyes closed and another minute of eyes open to bring the participants to a resting state. Later, 10 facial images of happiness, sadness, calm and fear from Radboud Faces Database (RafD) [18] are presented to the participants for 1 minute per emotion to elicit the corresponding emotional states. The stimuli presentation protocol is summarized in Figure 5.

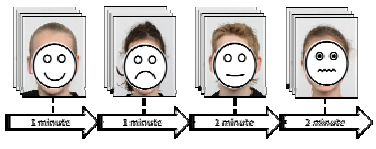
Duration (minutes)	Subject's states	Stimuli presentation
1	Eyes close	Null
1	Eyes open	Blank screen
4	Observing stimuli	

Figure 5: Stimuli presentation protocol

4.2 Signal Pre-Processing

EEG signals comprised of 5 frequency bands including delta (2.5 Hz to 4 Hz), theta (4Hz to 8 Hz), alpha (8 Hz to 13 Hz), beta (13 Hz to 30 Hz) and gamma (30 Hz to 60 Hz). In this study, signal pre-processing involves selecting frequency bands within 8 Hz to 60 Hz which includes alpha, beta and gamma bands. Signals in the lower frequency bands are discarded because normally such range represents inactive brain activity [35]. As a result, it produces the power spectral density of each channel, as depicted in Figure 6.

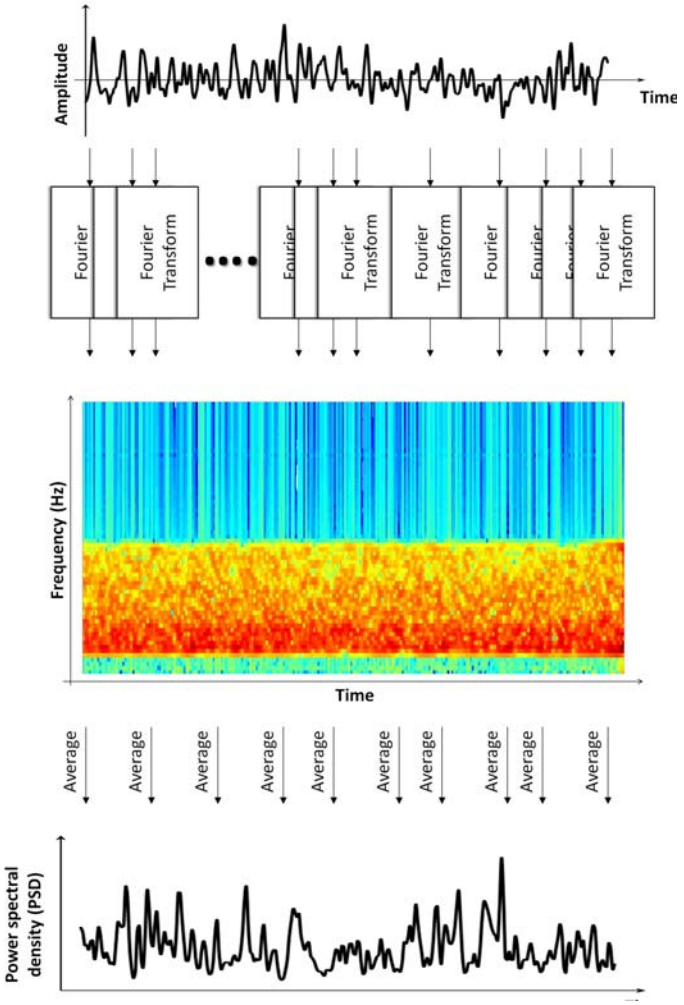


Figure 6: Generation of power spectral density (PSD) for frequency ranged from 8HZ to 60HZ (alpha, beta, theta) at channel C3 during elicitation of calm emotion by a subject

4.3 Feature Extraction

For this study, the proposed CMAC feature extraction techniques are compared with other existing techniques including Power Spectral Density (PSD) calculation, Kernel Density Estimation (KDE) and Mel Frequency Cepstral Coefficients (MFCC):

a) Kernel density estimation (KDE)

Kernel density estimation (KDE) is an approach for finding

the probability distribution function of random variables without having to assume the distributions (i.e., normal or skewed) of data samples. As a non-parametric approach, this estimation technique is also more flexible in the sense that the estimation is extracted directly from the data. Furthermore, KDE is able to portray a good representation of a continuous population [37].

For a given input with the sample point $(x_1, x_2, \mathrm{K}\ x_n)$ the estimate of the kernel density $\hat{f}(x)$ is defined as:

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where

K	kernel function, which is taken as normal distribution for our experiment
n	number of samples per frame
h	window width or bandwidth
K	kernel function, which is taken as normal distribution for our experiment
n	number of samples per frame
h	window width or bandwidth

The kernel *K* satisfies several conditions summarized in [37].

b) Mel Frequency Cepstral Coefficients (MFCC)

The Mel Frequency Cepstral Coefficients (MFCC) method for feature extraction. MFCC are commonly used as input features for solving pattern recognition and machine learning tasks. These coefficients are obtained by calculating the *mel frequency* bands which are ranged according to the frequency band of EEG signal. Although MFCC is more popular in speech processing, it is considered feasible for the processing of brainwaves [38].

4.4 Classification

Based on the dimensional outlook of emotion, *calm*, *fear*, *happiness* and *sadness* can be analyzed as the composites of valence and arousal in an affective space model, as depicted at the right most of Figure 3. The affective space model is constructed based on the generalization of quadrants containing each of the emotional states in the circumplex model of affect [34]. Hence, happiness lies on the positive valence and positive arousal quadrant. Fear is considered as an emotional state with negative valance and positive arousal. For the negative arousal, sadness and calm are placed at the negative and positive valence quadrants, respectively. Thus, for classification based on the supervised learning approach, positive cases are labeled as 1 and negative cases are labeled as -1.

In this study, Evolving Fuzzy Neural Network (EFuNN) was used to perform emotion classification. It is the first breed of evolving connectionist system (ECoS) complying with the seven requirements for an intelligent system [46]. As a variant of ECoS, structure of EFuNN is modified as the training instances are presented. Typically, the EFuNN architecture consists of five neuron layers as illustrated in Figure 7.

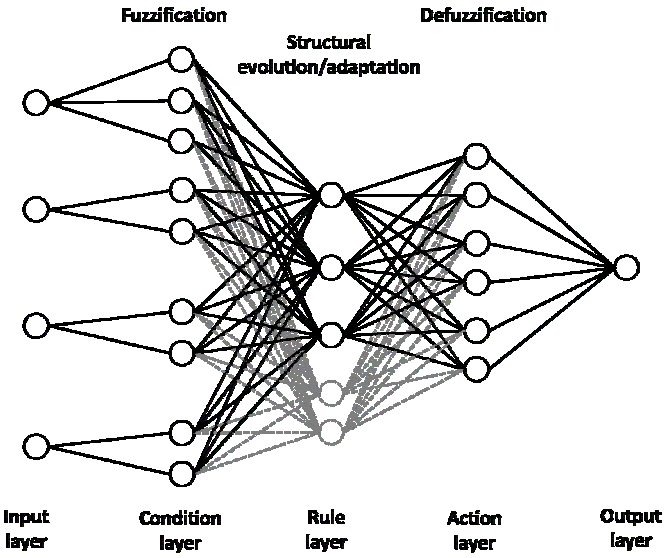


Figure 7: EFuNN architecture

The first layer is the input layer containing neurons of the same number of features in corresponding problem space. The second layer is the condition layer, in which fuzzification of input is performed based on the membership function defined by each neuron. Hence, neurons in this layer are not fully connected to the input layer, but each input neuron is only connected to its own subset of condition neurons. The third layer is the rule layer which evolves (grows and adapts) itself as the response to the incoming data. The fourth layer contains neurons of fuzzy membership functions that produce the fuzzy output values. In the fifth layer, the fuzzy outputs are defuzzified into the corresponding crisp outputs.

With that, two evolving fuzzy neural networks classifiers corresponding to valence and arousal are adapted. Each network consists of 8 input nodes representing the number of extracted features. The classifiers are evaluated based on several categories, as illustrated in Figure 8. For subject-dependent evaluation, a classifier is constructed and tested on the same subjects. On the other hand, for subject-dependent evaluation, the training set and the testing set are derived from entirely different subjects. Furthermore, a homogenous model is constructed based on the instances that are derived from only one subject. While for the heterogeneous case, the classifiers are trained from a certain number of subjects. In addition, for the memory testing, a model is tested using the same dataset that is used in the learning process. The result indicates how well the model is learned on the same instances. However, it does not consider such instances that are not included in the training set.

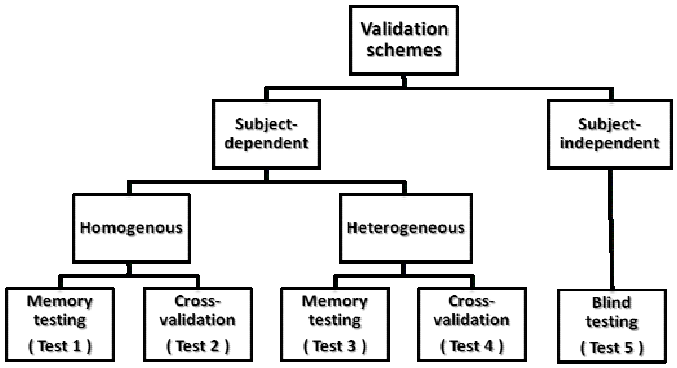


Figure 8: Validation schemes adapted in this study

5 Results

The results that are presented in Figure 9 show the average accuracy of emotional classification of 11 subjects using EFuNN on features from power spectral density, kernel density estimation, mel-frequency cepstral coefficients, as well as the CCMA (Type i).

Classification results of subject-dependent homogenous memory testing using EFuNN on features that are extracted based on the CCMA (Type I) and others are shown in Figure 9(a). The highest accuracy is obtained from MFCC features with a perfect classification. The minimum accuracy that is produced by MFCC features is 97.5%. S Slightly lower than that, classification accuracy that is obtained from features of CMAC weights ranges from 91.23% to 99.64%. Other than that, classification of KDE features and PSD features are averaged at 70.88% and 75.75%, respectively.

As shown in Figure 9(b), CCMA (Type I) produces the highest accuracy range in subject-dependent homogenous cross validation using EFuNN (between 77.4% and 99.52%). Compared to the previous case in Section 6.3.2.1, the classification accuracy from MFCC features drops tremendously from the previous highest accuracy of barely 60%. The lowest accuracy that is obtained by using MFCC features is 21.15%, which is only 2% higher than the lowest accuracy of KDE features. On the other hand, PSD features produce accuracy at the middle range from 33.17% to 74.04%. In short, it is observed that the classification performance of EFuNN on CCMA (Type I) is competitive with any other features on subject-dependent homogenous cross validation.

As mentioned, in subject-dependent heterogeneous memory testing, a training set is composed of instances of a certain number of subjects, which are also used for testing. In this research, heterogeneous testing is performed by using 6 subjects. Results of subject-dependent heterogeneous memory testing using EFuNN as the classifier are displayed in Figure 9(c). From the box plots, it is observed that classification of features from CCMA (Type I) produces the highest accuracy at 78.85%. That is followed by classification of MFCC features with classification accuracy between 54.33% and 69%. In addition, the classification of emotions using power spectral density and the classification of emotions using kernel density estimate features are observed to be below 50% with an average of 34.6% and 38.44%, respectively.

For subject-dependent heterogeneous cross validation, a subject is partitioned into 5 folds. With that, the training set is composed of 4 folds from 6 different subjects. Hence, the classification performance is determined based on the number of correctly identified instances in the remaining fold of each subject. Figure 9(d) presents the subject-dependent heterogeneous 5-fold cross validation using EFuNN. Based on the results, the best performance is obtained by the classification of emotions from CCMA (Type I) with the highest accuracy of 77.88%. The lowest accuracy from this feature is 44.71% which is still higher than the classification accuracy of other features. In relation to that, the performance of CMAC features is followed by features from power spectral density with the accuracy from 21.63% to 44.23%. Following that is MFCC features with classification accuracy between 19.23% and 43.18%. Classification of emotions using KDE features falls within the range of the accuracy obtained by MFCC features, which is from 22.12% to 29.33%.

Lastly, Figure 9(e) shows the classification accuracy of different features on subject-independent blind testing using EFuNN. The highest accuracy is obtained from MFCC features with 33.17% correctly classified instances. The lowest accuracy from that feature is 9.62%, which is also the lowest among the others. Despite that, the average accuracy for all features is very low (25.33%).

In addition, Figure 10 presents the classification results of the same subjects using features that are extracted using CCMA(Type I). It is indicated that subject-dependent homogenous memory classifications of 11 subjects produce accuracy between 91.23% and 99%. A wider range is observed on subject-dependent homogenous cross validation with the lowest accuracy of 77.4%.

For the heterogeneous case, classifying samples that are also used for the construction classifiers produces accuracy between 54.81% and 78.85%. Through cross validation, average classification accuracy from 44.71% to 77.88% is achieved. Similar to homogeneous, well performed classifications of heterogeneous cases can be useful for profiling emotions of a group of subjects. However, when classifying emotions using classifiers that are constructed based on samples of different subjects, the highest accuracy is barely 40%.

6 Conclusion

Overall, the classifications of emotions using features that are extracted using CMAC perform better than the classifications of power spectral density for subject-dependent homogenous and heteregeneous analysis. With that, the results support the viability for adapting such approach in other applications. One of the potential applications is the visualization of the underlying brain activations which currently can only be analyzed by experts in neurology. The resources are not only scarce but also complex. Because this model is applicable for brain signals that are captured using EEG, the costs of

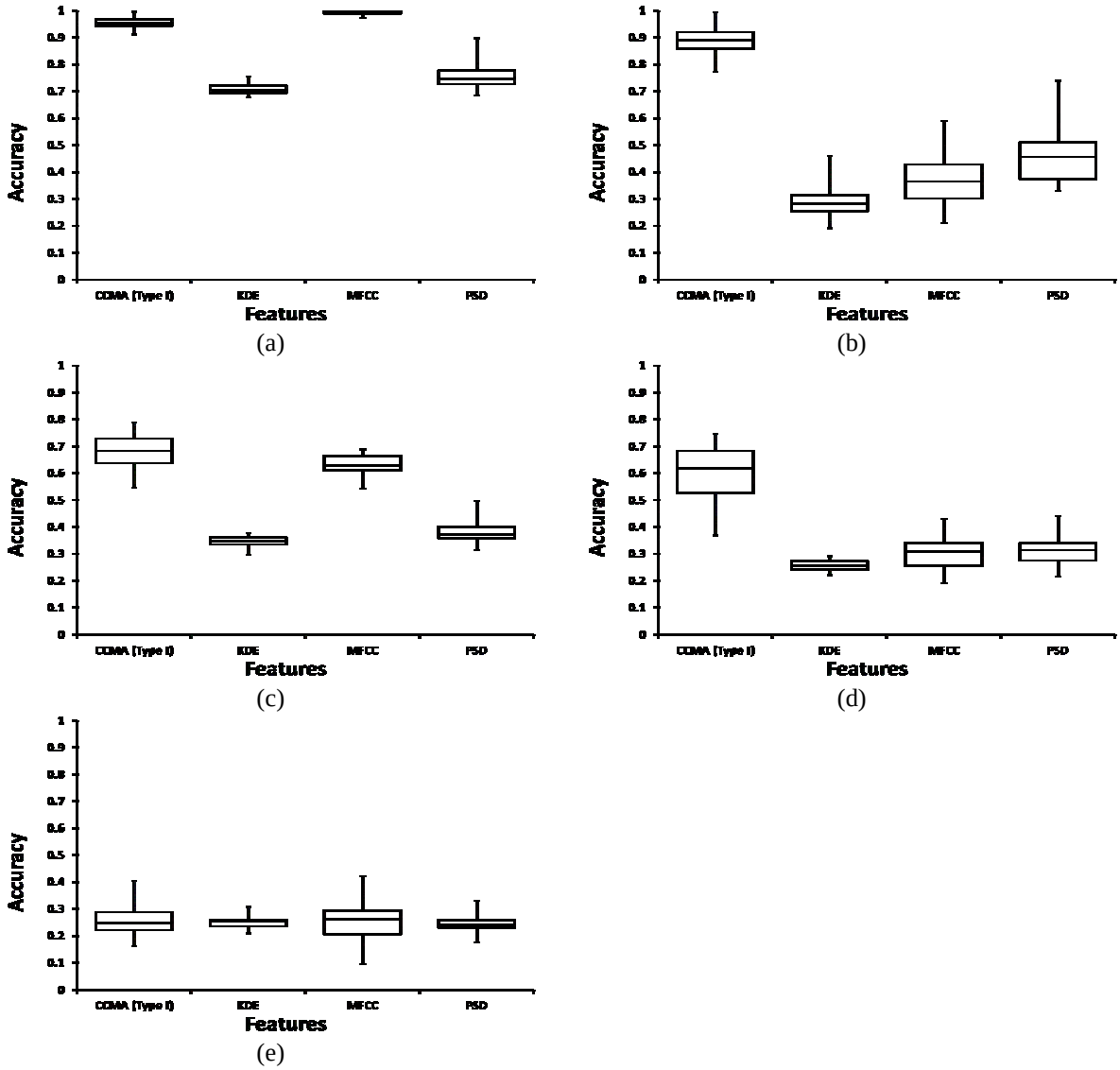


Figure 9: (a) Subject-dependent homogenous memory test, (b) Subject-dependent homogenous cross validation, (c) Subject-dependent heterogeneous memory test, (d) Subject-dependent heterogeneous croos validation, and (e) Subject-independent

employing other more expensive brain imaging tools can be reduced.

This is possible because the features are derived from the CMAC learning map which reflects the physical locations of the EEG electrodes positions. Thus, the visualization of the learning map may be analyzed to understand the underlyingneural correlates of brain activities. Going further, expanding the CMAC learning map into a 3-dimensional learning space is envisaged to produce a 3-dimensional brain model which can then estimate the source of the brain activations.

Furthermore, outstanding results of the homogenous classification can be useful for analyzing the dynamic of emotions on the corresponding subjects, as emotions do not occur abruptly [8] and changes over the course of time.

On the other hand, the classification performance for both features based on subject-independent blind tests are equally poor. The poor performance of subject-independent classification is not worse than the results that are reported in other studies [15, 24]. This is because the individual differences may influence the emotion processing [12] and can be captured at the brain [42].

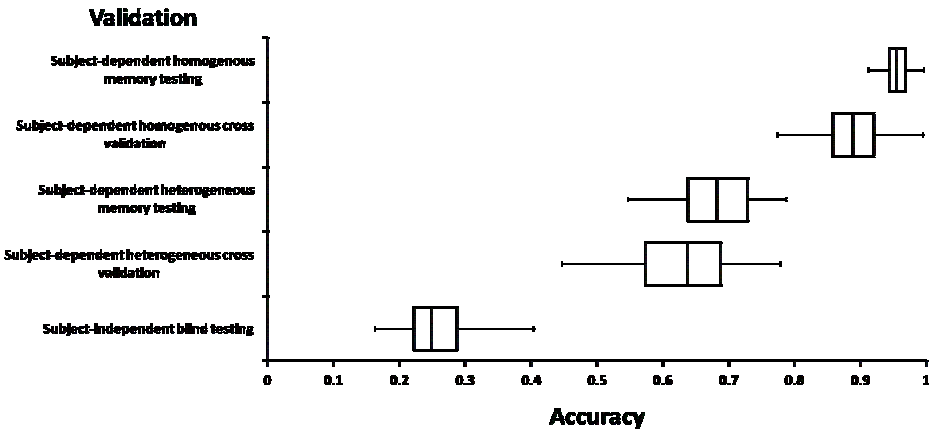


Figure 10: Classification performance of using CMAC-based features for different evaluation scenarios using EfuNN

References

[1] J. S. Albus, "A New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," *Journal of Dynamic Systems, Measurement, and Control*, doi:10.1115/1.3426922, 97(3):220, 1975.

[2] P. Bard and D. M. Rioch, "A Study of Four Cats Deprived of Neocortex and Additional Portions of the Forebrain," *Bulletin of the Johns Hopkins Hospital*, 60:73-153, 1937.

[3] T. F. Bastos-Filho, A. Ferreira, A. C. Atencio, S. Arjunan, and D. Kumar, "Evaluation of Feature Extraction Techniques in Emotional State Recognition," *4th International Conference on Intelligent Human Computer Interaction (IHCI)*, doi:10.1109/IHCI.2012.6481860, pp. 1-6, 2012.

[4] J. T. Cacioppo and G. G. Berntson, "Relationship between Attitudes and Evaluative Space: A Critical Review, with Emphasis on the Separability of Positive and Negative Substrates," *Psychological Bulletin*, doi:10.1037/0033-2909.115.3.401, 115(3):401-423, 1994.

[5] R. A. Calvo and S. D'Mello, "Affect Detection: An Interdisciplinary Review of Models, Methods, and Their Applications," *IEEE Transactions on Affective Computing*, doi:10.1109/T-AFFC.2010.1, 1(1):18-37, 2010.

[6] Tong Yuen Chai, Woo San San, Jee-Hou Ho, and M. Rizon, "Effectiveness of Statistical Features for Human Emotions Classification using EEG Biosensors," *Research Journal of Applied Sciences, Engineering and Technology*, *5*(21):5083-5089, 2013.

[7] G. Chanel, J. Kronegg, D. Grandjean, and T. Pun, "Emotion Assessment: Arousal Evaluation using EEG's and Peripheral Physiological Signals," *Multimedia Content Representation, Classification and Security*, Springer Berlin Heidelberg, pp. 530-537, 2006.

[8] T. Costa and M. Crini, "Basic Emotions: Differences in Time Sequence and Functional Imaging with Low Resolution Brain Electrical Tomography (LORETA)," *Nature Precedings*, Online: doi:10.1038/npre.2011.5566.1, p. 713, 2011.

[9] T. Dalgleish, B. D.Dunn, and D. Mobbs, "Affective Neuroscience: Past, Present, and Future," *Emotion Review*, doi:10.1177/1754073909338307, 1(4):355-368, 2009.

[10] A. Delorme and S. Makeig, "EEGLAB: An Open Source Toolbox for Analysis of Single-Trial EEG Dynamics Including Independent Component Analysis," *Journal of Neuroscience Methods*, doi:10.1016/j.jneumeth.2003.10.009, 134(1):9-21, 2004.

[11] P. Ekman, "Are There Basic Emotions?" *Psychological Review*, 99(3):550-553, (1992).

[12] L. T. Germine, and C. I. Hooker, "Face Emotion Recognition is Related to Individual Differences in Psychosis-Proneness," *Psychological Medicine*, doi:10.1017/S0033291710001571, 41(5):937-947, 2011.

[13] S. K. Hadjidimitriou and L. J. Hadjileontiadis, "EEG-Based Classification of Music Appraisal Responses Using Time-Frequency Analysis and Familiarity Ratings," *IEEE Transactions on Affective Computing*, doi:10.1109/T-AFFC.2013.6, 4(2):161-172, 2013.

[14] J. M. Harlow, *English: Recovery from the Passage of an Iron Bar through the Head, Journal Article about Phineas Gage.* Retrieved from http://commons.wikimedia.org/w/index.php?title=File:Recovery_from_the_passage_of_an_iron_bar_through_the_head.djvu&page=2, 1868.

[15] R. Horlings, D. Datcu, and L. J. M. Rothkrantz, "Emotion Recognition using Brain Activity," *Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, New York, NY, USA: ACM. doi:10.1145/1500879.1500888, pp. 6:II.1–6:1, 2008.

[16] R. Khosrowabadi, A. Wahab, K. K. Ang, and M. H. Baniasad, "Affective Computation on EEG Correlates of Emotion from Musical and Vocal Stimuli," *IJCNN 2009*,

International Joint Conference on Neural Networks, Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5178748, pp. 1590-1594, 2009.

[17] H. Korn and P. Faure, "Is There Chaos in the Brain? II. Experimental Evidence and Related Models," *Comptes Rendus Biologies*, 326(9):787-840, 2003.

[18] O. Langner, R. Dotsch, G. Bijlstra, D. H. J. Wigboldus, S. T. Hawk, and A. van Knippenberg, "Presentation and Validation of the Radboud Faces Database," *Cognition and Emotion*, 24(8):1377-1388, 2010.

[19] M. Li, Q. Chai, T. Kaixiang, A. Wahab, and H. Abut, "EEG Emotion Recognition System," K. Takeda, H. Erdogan, J. H. L. Hansen, and H. Abut (Eds.), *Vehicle Corpus and Signal Processing for Driver Behavior*, Springer US, Retrieved from http://link.springer.com/chapter/10.1007/978-0-387-79582-9_10, pp. 125-135, 2009.

[20] M. Li and B.-L. Lu, "Emotion Classification Based on Gamma-Band EEG," *Proceedings: Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, doi:10.1109/IEMBS.2009.5334139, 2009:1323-1326, 2009.

[21] Y. P. Lin, C. H. Wang, T. P. Jung, T. L. Wu, S. K. Jeng, J. R. Duann, and J. H. Chen, "EEG-Based Emotion Recognition in Music Listening," *Biomedical Engineering, IEEE Transactions*, 57(7):1798-1806, 2010.

[22] Y. Liu, O. Sourina, and M. K. Nguyen, "Real-Time EEG-Based Human Emotion Recognition and Visualization," *2010 International Conference on Cyberworlds (CW)*, doi:10.1109/CW.2010.37, pp. 262-269, 2010.

[23] M. Macas, M. Vavrecka, V. Gerla, and L. Lhotska, "Classification of the Emotional States Based on the EEG Signal Processing," *9th International Conference on Information Technology and Applications in Biomedicine, 2009, ITAB 2009* doi:10.1109/ITAB.2009.5394429, pp. 1-4, 2009.

[24] E. T. Mampusti, J. S. Ng, J. J. I. Quinto, G. L. Teng, M. T. C. Suarez, and R. S. Trogo, "Measuring Academic Affective States of Students via Brainwave Signals," *2011 Third International Conference on Knowledge and Systems Engineering (KSE)*, doi:10.1109/KSE.2011.43, pp. 226-231, 2011.

[25] A. Mehrabian, "Pleasure-Arousal-Dominance: A General Framework for Describing and Measuring Individual Differences in Temperament," *Current Psychology*, 14(4):261-292, 1996.

[26] M. Molavi, J. Bin Yunus, and E. Akbari, "Comparison of Different Methods for Emotion Classification," *Modelling Symposium (AMS), 2012 Sixth Asia*, doi:10.1109/AMS.2012.53, pp. 50-53, 2012.

[27] M. Murugappan, M. Rizon, R. Nagarajan, S. Yaacob, I. Zunaidi, and D. Hazry, "Lifting Scheme for Human Emotion Recognition using EEG," *International Symposium on Information Technology, 2008. ITSim 2008*, doi:10.1109/ITSIM.2008.4631646, 2:1-7, 2008.

[28] E. Niedermeyer and F. H. L. da Silva, *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields*, Lippincott Williams & Wilkins, 2005.

[29] J. Panksepp, *Affective Neuroscience: the Foundations of Human and Animal Emotions*, Oxford University Press, 2004.

[30] P. C. Petrantonakis and L. J. Hadjileontiadis, "Emotion Recognition from Brain Signals using Hybrid Adaptive Filtering and Higher Order Crossings Analysis," *IEEE Transactions on Affective Computing*, doi:10.1109/T-AFFC.2010.7, 1(2):81-97, 2010.

[31] P. C. Petrantonakis and L. J. Hadjileontiadis, "A Novel Emotion Elicitation Index using Frontal Brain Asymmetry for Enhanced EEG-Based Emotion Recognition," *IEEE Transactions on Information Technology in Biomedicine: A Publication of the IEEE Engineering in Medicine and Biology Society*, doi:10.1109/TITB.2011.2157933, 15(5):737-746, 2011.

[32] R. Plutchik, *Emotions and Life: Perspectives from Psychology, Biology, and Evolution* (1st ed.), American Psychological Association (APA), 2002.

[33] K. S. Rahnuma, A. Wahab, N. Kamaruddin, and H. Majid, "EEG Analysis for Understanding Stress Based on Affective Model Basis Function," *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)*, doi:10.1109/ISCE.2011.5973899, pp. 592-597, 2011.

[34] J. A. Russell, "A Circumplex Model of Affect," *Journal of Personality and Social Psychology*, doi:10.1037/h0077714, 39(6):1161-1178, 1980.

[35] S. Sanei and J. A. Chambers, *EEG Signal Processing* (1st ed.), Wiley-Interscience, 2007.

[36] K. R. Scherer, "What Are Emotions? And How Can They Be Measured?" *Social Science Information*, doi:10.1177/0539018405058216, 44(4):695-729, 2005.

[37] X. Shen and S. Agrawal, "Kernel Density Estimation for an Anomaly Based Intrusion Detection System, *Proceedings of the International Conference on Machine Learning; Models, Technologies and Applications*, Las Vegas, NV, pp. 161-167, 2006.

[38] M. Slaney, Auditory Toolbox Version Two, Interval Research Corporation, No. 1998-010, pp. 1-41, 1998.

[39] M. Soleymani, M. Pantic and T. Pun, "Multimodal Emotion Recognition in Response to Videos," *IEEE Transactions on Affective Computing*, doi:10.1109/T-AFFC.2011.37, 3(2):211-223, 2012.

[40] O. Sourina, Y. Liu and M. K. Nguyen, "Emotion-Enabled EEG-based Interaction," *SIGGRAPH Asia 2011 Posters*, New York, NY, USA: ACM. doi:10.1145/2073304.2073315, pp. 10:1-10:1, 2011.

[41] C. J. Stam, "Nonlinear Dynamical Analysis of EEG and MEG: Review of an Emerging Field," *Clinical Neurophysiology: Official Journal of the International Federation of Clinical Neurophysiology*, doi:10.1016/j.clinph.2005.06.011, 116(10):2266-2301, 2005.

[42] J. S. Stevens and S. Hamann, "Sex Differences in Brain

Activation to Emotional Stimuli: A Meta-Analysis of Neuroimaging Studies," *Neuropsychologia*, doi:10.1016/j.neuropsychologia.2012.03.011, 50(7):1578-1593, 2012.

[43] S. D. Teddy, C. Quek and E. K. Lai, "PSECMAC: a Novel Self-Organizing Multiresolution Associative Memory Architecture," *IEEE Transactions on Neural Networks / a Publication of the IEEE Neural Networks Council*, doi:10.1109/TNN.2007.912300, 19(4):689-712, 2008.

[44] S. S. Tomkins, *Affect Imagery Consciousness: The Complete Edition*, Springer Publishing Company, 2008.

[45] K. C. Tseng, B.-S. Lin, C.-M. Han and P.-S. Wang, "Emotion Recognition of EEG Underlying Favourite Music by Support Vector Machine," *2013 International Conference on Orange Technologies (ICOT)* doi:10.1109/ICOT.2013.6521181, pp. 155-158, 2013.

[46] M. J. Watts, "A Decade of Kasabov's Evolving Connectionist Systems: A Review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, doi:10.1109/TSMCC.2008.2012254, 39(3):253-269, 2009.

[47] H. J. Yoon and S. Y. Chung, "EEG Spectral Analysis in Valence and Arousal Dimensions of Emotion," *2011 11th International Conference on Control, Automation and Systems (ICCAS)*, pp. 1319-1322, 2011.

Hamwira Yaacob received the B.Sc. degree from University of the Pacific, Stockton, California USA, in 1999, and the M.Sc. in Intelligent Systems degree from University Utara Malaysia, in 2006. He is currently a lecturer at the International Islamic University Malaysia. His current research interests include computational intelligence, machine learning, affective computing, and neuro-cognitive computing, in which a number of conference papers and book chapter have been authored and several research grants have been secured.

Abdul Wahab received the Degree from the University of Essex, Essex, U.K., in 1979, the M.Sc. degree from the National University of Singapore, Singapore, in 1987, and the Ph.D. degree from Nanyang Technological University, Singapore. His research has been in the areas of telecommunication, signal processing, and artificial intelligence. He was with Hewlett Packard Singapore, Singapore, as a Research and Development Project Manager both in CO, USA. He joined Nanyang Technological University in 1990, where he was an Associate Professor, before joining the International Islamic University of Malaysia, Malaysia, as a Professor, in 2009. He has authored over 100 conference papers, journal papers, patents, and book chapters in the areas of digital and optical computing, signal processing, and artificial intelligence.

Norhaslinda Kamaruddin currently holds a post of Senior Lecturer in Faculty of Computer and Mathematical Sciences, MARA University of Technology, Malaysia since 2011. She received her Bachelor degree in Information Technology (Computer Science) from Universiti Kebangsaan Malaysia in 2001 followed by her Master of Software Engineering from Malaya University in 2006. In 2013, she is awarded with Doctor of Philosophy (Computer Engineering) from Nanyang Technological University (Singapore) focusing on computational intelligence. She is very active in research fields of affective computing, speech emotion recognition, neuro-cognitive informatics and driver behavioral study.

An Approach to Dynamically Adjusting Clusters and Outliers for Multi-Dimensional Data Sets

Yong Shi*
Kennesaw State University, Kennesaw, GA 30144 USA

Abstract

Nowadays a large amount of data are generated in many research and industry fields, such as bioinformatics, social media exchange, meteorology, etc. Many of the real data sets contain dynamic contents. In this paper, we discuss the data distribution of data sets that change constantly. We analyze the change of the distribution in multi-dimensional data space, and propose an approach to efficiently and effectively process the multi-dimensional data sets. Real data sets contain outliers and clusters. Outlier detection is concerned with discovering the exceptional behaviors of certain objects. Clustering algorithms separate data points into different groups, in a way that data points in the same group have high similarity and data points from different groups are different from each other. We analyze the meaning of clusters and outliers and propose an algorithm to dynamically change the set of clusters and the set of outliers as the data set changes. The reconstructed clusters and outliers can help improve the performance of data analysis such as nearest neighbor search and dimension reduction.

1 Introduction

Cluster and outlier detection has always been one of the focuses of data mining research. Outlier detection is an important branch in the field of data mining with numerous applications, such as credit card fraud detection, discovery of criminal activities, discovery of computer intrusion, data cleaning, network intrusion detection, clinical diagnosis of diseases, etc. An outlier is a data point that does not follow the main characteristics of the input data [15]. Outlier detection is concerned with discovering the exceptional behaviors of certain objects. There are numerous studies on outlier detection. E. M. Knorr etc. [6] detected a distance-based outlier. Ramaswamy etc. [14] further extended it based on the distance of a data point from its *k*th nearest neighbor. Orair etc. [10] assess distance-based outlier detection approaches and evaluate them, with the conclusion that the combination of optimization strategies enables significant gains. Last etc. [7] apply the fuzzy set theory to model the human perception of exceptional values in order to find outliers in a data set.

Cluster analysis is used to identify homogeneous and well-separated groups of objects in data sets. It provides a way to learn about the structure of complex data and acquire useful information from data sets. In clustering, some details are disregarded to simplify the data. Clustering can be viewed as concise summaries of the data, and it is related to many disciplines and plays an important role in a broad range of applications which deal with large data sets and data with many attributes. Automatic cluster detection is an important tool for unsupervised knowledge discovery. Many methods have been proposed in the literature. Well known approaches include partitioning, hierarchical, grid-based, density-based algorithms, and more [4, 8] . For example, grid-based algorithms divide the space into grids and perform all operations on the grids to improve the processing time, no matter what the size of a data set. Partitioning algorithms requires K as the input information, which is the number of clusters, and gradually partition the data sets into K groups. Hierarchical algorithms present the clusters on different levels of details, helping users decide the scale of the clusters they need. Density-based algorithms decide the membership of each data point in a data set based on how close it is to a certain data point. All existing clustering algorithms have advantages and disadvantages.

A lot of the algorithms do not perform well when the dimensional goes higher. One of the solutions to this problem is dimension reduction which tries to find sub-clusters in a low-dimensional subspace. For example, Kim etc. [5] adopt dimension reduction methods to reduce the dimension of the document vectors. In order to handle the classification problem where a document may belong to multiple classes, they design decision functions for the centroid-based classification algorithm and support vector classifiers. Ding etc. [3] design an approach which combines linear discriminant analysis and K-means clustering to select the most discriminative subspace, and generate class labels simultaneously. Boulesteix [2] compares the classification procedure consisting of Partial Least Squares dimension reduction and linear discriminant analysis with well-known classification methods, and proposes a procedure to choose the number of Partial Least Squares components.

Many algorithms have been presented to process data sets

*Department of Computer Science. Email: yshi5@kennesaw.edu. 1000 Chastain Road.

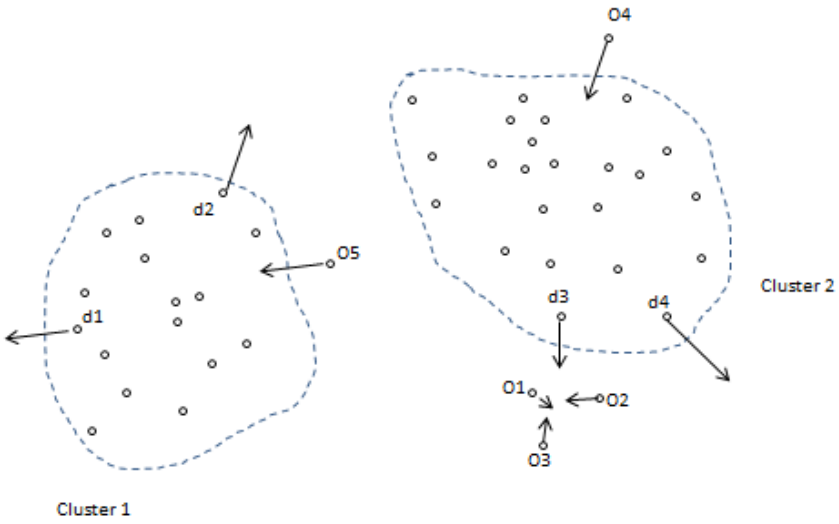


Figure 1: A dynamic data set with clusters and outliers a two dimensional data space

with constant changes. These algorithms use various strategies to efficiently and effectively keep track of the change of the data sets, and adjust the cluster information dynamically to keep it consistent with the data change. Some of them use certain motion models to track clusters, and others use dissimilarity measurement and provide dynamic procedures of splitting and merging. For example, Peng etc. [12] introduce class rate, based on which a dynamic clustering algorithm is designed to divide the entire data set into all possible classes, with the clustering strategy which results in the minimum class ratio being chosen. Papadogiannis etc. [11] present a dynamic clustering approach for the formation of the clusters of cooperating base stations for a cellular network. It incorporates multi-cell cooperative processing, which uses linear beamforming for the sum-rate maximization of the uplink. Omran etc [9] present a dynamic clustering algorithm for image segmentation, which automatically determines the number of clusters and simultaneously clusters the data set with minimal user interference. Qing etc. [13] introduce a distributed energy-efficient clustering scheme for heterogeneous wireless sensor networks, which selects the clusters based on the ratio between residual energy of each node and the average energy of the network.

2 Dynamically Adjusting Clusters and Outliers

Figure 1 shows an example of a two dimensional data set. This data set contains two clusters: cluster1 and cluster2. It also contains numerous outliers: O1, O2, O3, O4, and O5. A lot of data points in this data set change their positions constantly. The arrows in the figure show the direction of the movement of some data points. From the figure, we can see that data points d1 and d2 are moving out of cluster1, outlier O5 is moving towards the centroid of cluster1, and will soon become a new data point of cluster1. Data points d3 and d4 are moving out of cluster2, and outlier O4 is moving into cluster2 to become a part of it. Outliers O1, O2, O3 are moving in different directions, but their movement makes them become closer and closer, possibly forming a new cluster. Data point d3 is also moving towards them, and it may become a new member of the new cluster as well. From Figure 1 we can see that there are many different situations in terms of the movement of data points within a data set, so it is crucial to carefully analyze different movement, and make adjustment of the data distribution accordingly.

When dimensionality changes, the movement of the data points, the relationship between clusters and the relationship between outliers can all be changed. Figure 2 shows an example of a three dimensional data set. It has a cluster named cluster2 that appears similarly to the one in Figure 1. However, since the dimensionality changes, the direction d3 is moving to might not be the same direction outliers O1, O2, O3 are moving to anymore, so d3 will not become a new member of a new cluster O1, O2, O3 will form.

Based on the observation and discussion above, we propose an approach to clustering and outlier detection problems. We analyze data sets with constant changes, and keep track of the status of clusters and the movement of data points, as well as the updated group of outliers. Different from the traditional approaches which are focused on two-dimensional or low-dimensional data spaces, we aim to analyze data sets in multi-dimensional data spaces. We also propose to adjust the clusters and outliers simultaneously, since they are two concepts that are closely related.

Before the discussion of our approach we will first introduce a few notations and definitions. Let n denote the total number of data points and d be the dimensionality of the data space. Let D_l be the lth dimension, where l = 1, 2, ..., d. Let the input d-dimensional data set DS contain $\mathbf{X}$

$$\mathbf{X} = \{X_1, X_2, ..., X_n\}, \quad (1)$$

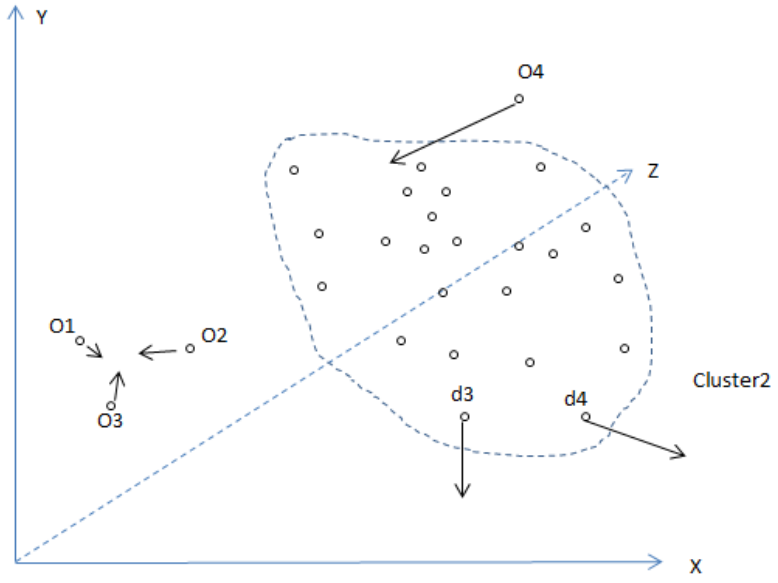


Figure 2: A dynamic data set with clusters and outliers in a three dimensional data space

which is normalized to be within the hypercube $[0,1]^d \subset R^d$. Each data point X_i (i=1,2,...,n) is a d-dimensional vector:

$$X_i = [x_{i1}, x_{i2}, ..., x_{id}]. \quad (2)$$

Data point X_i has the *id* number i.

Given a data set DS, we will first generate original clusters and outliers. Any existing approach can be used to acquire the original set of clusters and the original set of outliers.

Let the original number of clusters be k_c, and the original number of outliers be k_o. Let the set of clusters be $\mathscr{C} = \{C_1, C_2, ..., C_{k_c}\}$, and the set of outliers be $\mathscr{O} = \{O_1, O_2, ..., O_{k_o}\}$. Figure 3 shows an example of a data set with a group of clusters and a group of outliers in a two dimensional data space. We can see that in cluster C_1, data points d1, d2, d3 are moving out of the cluster. Data point d1 is moving towards C_3, and might become a new member of C_3 soon; data point d3 is moving towards C_2, and might become a new member of C_2 soon; data point d2 is moving and soon will become a new outlier. In cluster C_2, data point d4 is moving towards C_3, and might become a new member of C_3 soon. In cluster C_3, data point d5 is moving towards C_4, and might become a new member of C_4 soon. In cluster C_4, data point d6 is moving towards C_3, and might become a new member of C_3 soon. In cluster C_{k_c}, data points d7, d8, d9 are moving out of the cluster. Data point d7 is moving towards C_4, and might become a new member of C_4 soon; data point d8 is moving towards C_3, and might become a new member of C_3 soon; data point d9 is moving and soon will become a new outlier. Outlier O1, O2, O3 are moving towards a direction that may form a new cluster. Outlier O4 is moving towards C_3, and might become a new member of C_3 soon. Outlier O_{k_o} is moving, but will not become a new member of a cluster soon. Since quite a few data points are moving out for both C_1 and C_{k_c}, these two clusters might be dismissed in the near future when they no longer contain enough data points. A lot of data points and outliers are moving towards C_3, and C_3 might soon contain more than half of the data points in the data set. It then needs to be split, in order to keep the balance of the cluster size. The number of data points in C_2 and C_4 do not change significantly, so they might stay the same.

In order to keep track of the relationship between dynamically moving data points and clusters, for each cluster $C_i \in \mathscr{C}$, i=1,2,...,k_c, we define its size. Let D_k be the kth dimension, where k = 1, 2, ..., d. Let p_k and q_k be the lower bound and upper bound of the value range on D_k , k = 1, 2, ..., d. We define the size of a cluster C as:

$$d\_size(C) = ((q_1 - p_1) * (q_2 - p_2) * ... * (q_d - p_d))^{1/d}, \quad (3)$$

When the dimensionality goes higher, we only need to recalculate the average size of C on each dimension. This design avoids the problem of increasing the volumes of the clusters nonlinearly when the dimensionality goes higher.

We also define the centroid of C as:

$$centroid(C) = (\frac{q_1 + p_1}{2}, \frac{q_2 + p_2}{2}, ..., \frac{q_d + p_d}{2}), \quad (4)$$

For a given data point DP, if $d(DP, centroid(C)) <= d\_size(C)/2$, DP is in C.

We also define the "relationship" between a cluster C and an outlier O as:

$$rel(C, O) = \frac{d(centroid(C), O)}{d\_size(C)}, \quad (5)$$

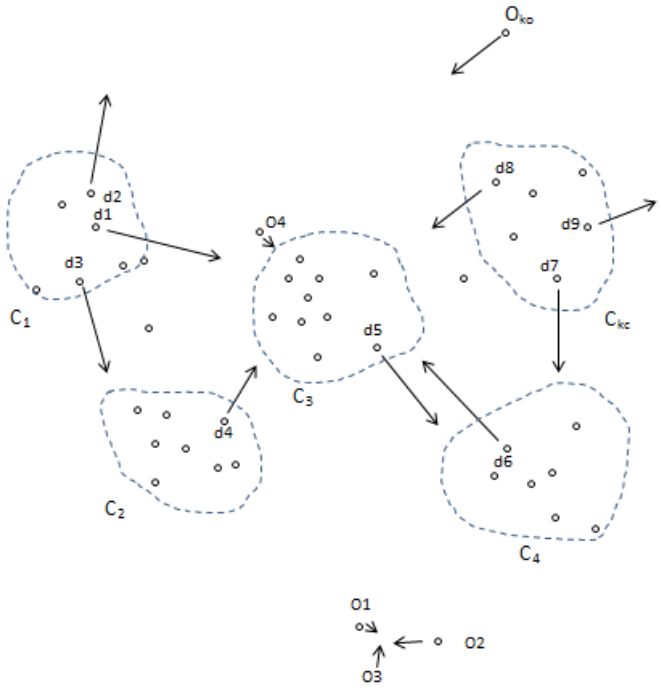


Figure 3: A data set with a group of clusters and a group of outliers in a two dimensional data space

In our approach, we closely keep track of the change of clusters and outliers based on the movement of the data points in a data set DS. Based on how fast the data points in DS change their positions and availabilities, a time interval t is assigned to DS.

At each time interval t:

Step 1) For each cluster $C_i \in \mathscr{C}$, we check the change of position for each data point $X_p \in C_i$, and count the number n_i of data points in C_i whose new value(s) on a certain dimension or certain dimensions are no long within $d\_size(C)$. If a data point no longer exists in DS, i.e., it is deleted from DS, it will also be counted into n_i.

If n_i exceeds a certain threshold, we will modify the size $d\_size(C_i)$ so it will still contain the majority of data points in C_i. If n_i does not exceed the threshold, they should be removed from C_i.

Step 2) For each outlier O_j in $\mathscr{O}$, we first find the minimum m_j of rel(C_i, O_j), i=1,2,...,k_c. If m_j is less than a threshold T1, O_j will be a new data point of C_i; otherwise, we find its neighbors. If O_j has enough neighbors, we will form a new cluster $C_j\prime$ that contains O_j and all its neighbors.

Step 3) If there are two clusters C_i and C_j, $d\_size(C_i) + d\_size(C_j) >= d(centroid(C_i, C_j)$, that means these two clusters are really close, or they even overlap each other. In this case, we need to merge C_i and C_j, and form a new cluster $C_{ij} = C_i \bigcup C_j$.

Step 4) If there is a cluster C_i, so that the number of the data points in C_i is less than a threshold T, we need to dismiss C_i, and its data points become new outliers.

Step 5) If there is a cluster C_i, so that the number of the data points in C_i is larger than half of the data set size, we need to split C_i into C_{i1} and C_{i2}, in order to keep the clusters balanced. Only the largest cluster in the entire cluster set possibly contains the number of data points larger than half of the data set size. We find two data points DP_k and DP_m so that d(DP_k , DP_m) is the largest among all the possible pairs of data points in C_i. C_{i1} contains DP_k, and C_{i2} contains DP_m. Data points closer to DP_k will be contained in C_{i1}, and Data points closer to DP_m will be contained in C_{i2}.

3 Algorithm

In this section we present our algorithm to adjust the clusters and outliers as a given data set constantly changes.

Figure 4 presents our algorithm. Given a data set DS, we first generate the original cluster set and outlier set. We then define a time interval t based on the frequency of change of the dynamic data set. For each interval, we modify the clusters and outliers as described in the previous section.

At each interval, we calculate the size of each cluster, and then calculate rel(C_i, O_j) for each pair of cluster and outlier. Next we adjust the size of clusters, and decide if we need to include an outlier in a cluster. We also check if we should merge two clusters, as well as if we should dismiss some clusters. We also check if a cluster becomes too big, in which case we need

Algorithm Dynamic Data Processing (*DS: data set, D: dimensions, the original cluster set* $\mathscr{C} = \{C_1, C_2, ..., C_{k_c}\}$, *the original outlier set* $\mathscr{O} = \{O_1, O_2, ..., O_{k_o}\}$)
Begin
1) Define a time interval t based on the frequency of change for positions and availabilities of data points in DS;
2) Monitor and record the change of data points' positions and availabilities dynamically;
3) At each interval t,
a) for each cluster $C_i \in \mathscr{C}$, i=1,2,...,k_c, we calculate its size $d\_size(C_i)$, and centroid of C_i: centroid(C_i);
b) calcluate rel(C,O) for each cluster C and each outlier O;
c) check if we should change $d\_size(C_i)$ for i=1,2,...,k_c;
d) check if an outlier O_j should be included in a cluster, or should form a new cluster with its neighbors;
e) check if two clusters C_i and C_j should be merged;
f) check if a cluster C_i, i=1,2,...,k_c, should be dismissed;
g) check if we need to split the largest cluster in $\mathscr{C}$;
5) Keep performing the algorithm until the data set no longer changes or the user interrupts the process;
End.

Figure 4: Proc: dynamically adjusting clusters and outliers

to split it into two smaller ones to keep the balance of the data distribution.

4 Experiments

Experiments are run on Intel(R) Pentium(R) 4 with CPU of 3.39GHz and Ram of 0.99 GB, to test our algorithm. We use both synthetic data sets and real data sets.

A synthetic data generator is designed for the experiments. The generator produces data sets with normalized distributions. The sizes of the data sets vary from 1,000, 2,000, ... to 9,000, with the gap of 1,000 between each two adjacent data set sizes. The dimensions of the data sets vary from 3, 6, ... to 60, with the gap of 3 between each two adjacent numbers of dimensions.

Figure 5 shows, as time goes on, how the number of the clusters in a synthetic 5-dimensional data set changes. This data set contains 2000 data points.

Figure 6 shows, as time goes on, how the number of the outliers in a synthetic 5-dimensional data set changes. This data set contains 2000 data points.

Figure 7 shows, as time goes on, how the average size of the clusters in a synthetic 5-dimensional data set changes. This data set contains 2000 data points.

We next evaluate the effectiveness of our proposed approach on real data sets. We obtained the real data sets from UCI Machine Learning Repository [1]. Several real data sets are selected including: 1) Glass data set containing 214 data points with 9 dimensions. There are 7 classes in the glass data, class 1 to class 7; 2) Wine Recognition data set containing 178 instances. The dimensionality of the data set is 13. It contains three natural clusters with the sizes of 59, 71 and 48; 3) Iris data set containing 150 data points with 4 dimensions. There are 3 classes in the iris data: Iris-setosa, Iris-versicolor, and Iris-virginica; 4) Ecoli data set containing 8 clusters and 336 instances, each of which having 7 features; 5) Ionosphere data set containing 351 data points with 34 dimensions. There are two classes in the ionosphere data: g as good, and b as bad.

We use *precision* and *recall* to measure the accuracy of a detected cluster. For a cluster C_i^s detected by an algorithm and a real cluster C_i^o, we define the precision of C_i^s with respect to C_i^o as $\frac{|C_i^s \bigcap C_i^o|}{|C_i^s|}$ and the recall as $\frac{|C_i^s \bigcap C_i^o|}{|C_i^o|}$. We choose the C_i^s as a corresponding cluster of C_i^o if the precision and recall of C_i^s with respect to C_i^o are high.

We compare the accuracy of our algorithm with existing algorithms such as CURE. As Table 1 indicates, the average accuracy rate of our algorithm is 90.16%, which is higher than the accuracy rate of CURE (89.33%).

5 Conclusion and Discussion

In this paper, we present an approach to dynamically adjusting clusters and outliers as a data set constantly changes. We change the content and size of clusters and outliers based on various conditions. The reconstructed clusters and outliers can help improve the effectiveness and efficiency of future data processing such as similarity search and dimension reduction.

References

[1] S. D. Bay, The UCI KDD Archive [http://kdd.ics.uci.edu], University of California, Irvine, Department of Information and Computer Science, 2013

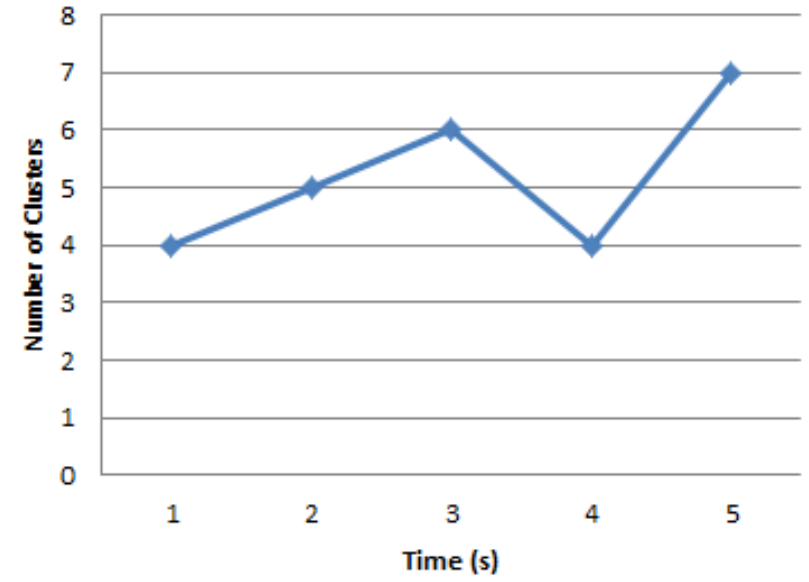


Figure 5: Change of cluster numbers as time goes on

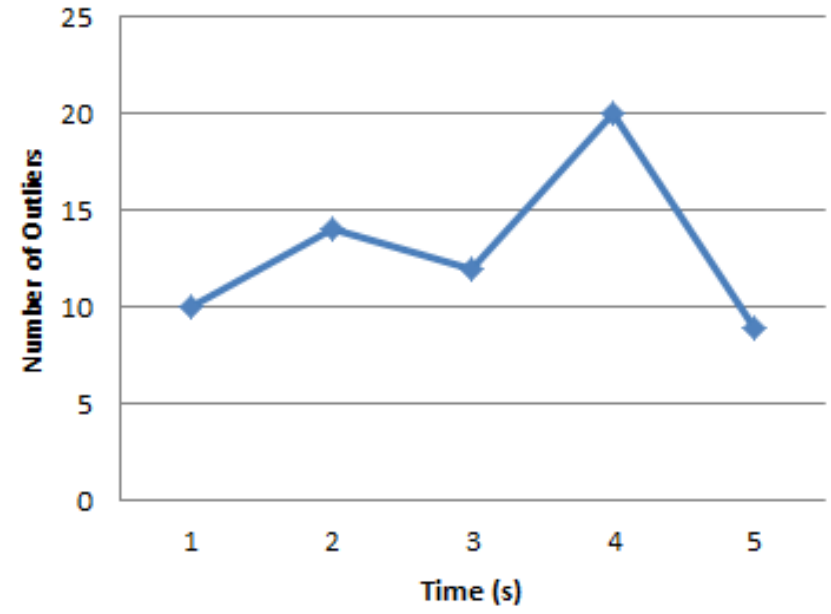


Figure 6: Change of outlier numbers as time goes on

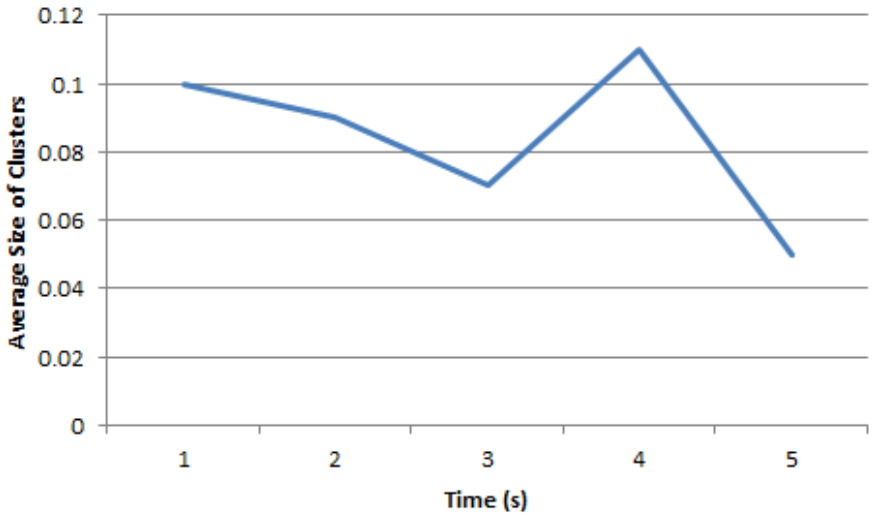


Figure 7: Change of average cluster size as time goes on

Table 1: Comparison of clustering result of CURE and out approach for real data sets

	CURE	Our approach
Average precision(%)	89.12	90.09
Average recall(%)	89.54	90.23

[2] A. Boulesteix, "PLS Dimension Reduction for Classification with Microarray Data," Statistical Applications in Genetics and Molecular Biology 3, 1(33):1–33, 2004.

[3] C. Ding and T. Li, "Adaptive Dimension Reduction Using Discriminant Analysis and K-means Clustering," ICML '07, Corvalis, Oregon, USA, pp. 521–528, 2007.

[4] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, 1990.

[5] H. Kim, P. Howland, and H. Park, "Dimension Reduction in Text Classification with Support Vector Machines," J. Mach. Learn. Res., 6:37–53, 2005.

[6] E. Knorr and R. Ng, "Algorithms for Mining Distance-Based Outliers in Large Datasets," *Proceedings of the 24th VLDB Conference*, New York, NY, USA, pp. 392-403, 1998.

[7] M. Last and A. Kandel "Automated Detection of Outliers in Real-World Data," *Proc. of the Second International Conference on Intelligent Technologies*, pp. 292–301, 2001.

[8] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Volume I, Statistics, 1967.

[9] M. Omran, A. Salman, and A. Engelbrecht, "Dynamic Clustering using Particle Swarm Optimization with Application in Image Segmentation," Pattern Analysis and Applications, 8:332–344, 2006.

[10] G. Orair, C. Teixeira, W. Meira, Y. Wang and S. Parthasarathy, "Distance-based Outlier Detection: Consolidation and Renewed Bearing," *Proc. VLDB Endow*. 3:1469–1480, 2010.

[11] A. Papadogiannis, D. Gesbert, and E. Hardouin, "A Dynamic Clustering Approach in Wireless Networks with Multi-Cell Cooperative Processing," ICC '08, Beijing, China, pp. 4033-4037, 2008.

[12] T. Peng, M. Jiang, and M. Hu, "A Dynamic Clustering Algorithm Based on Small Data Set," CGIV '09, Tianjin, China, pp. 410-413, 2009.

[13] L. Qing, Q. Zhu, and M. Wang, "Design of a Distributed Energy-efficient Clustering Algorithm for Heterogeneous Wireless Sensor Networks ," Computer Communications, pp. 29:2230–2237, 2006.

[14] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient Algorithms for Mining Outliers from Large Data Sets," Proceedings of the ACM Sigmod Conference on Management of Data, Dallas, Texas, USA, pp. 427–438, 2000.

[15] M. Wu and C. Jermaine, "Outlier Detection by Sampling with Accuracy Guarantees," KDD '06, New York, NY, USA, pp. 767-772, 2006.

Yong Shi received the BS and MS degrees, both in Computer Science, from the University of Science and Technology of China in 1996 and 1999, respectively. He received Ph.D. in Computer Science from the State University of New York at Buffalo in 2006. He is currently an Associate Professor in the Department of Computer Science in Kennesaw State University. His research interests include data mining, database, machine learning, and information retrieval.

Instructions For Authors

The International Journal of Computers and Their Applications is published multiple times a year with the purpose of providing a forum for state-of-the-art developments and research in the theory and design of computers, as well as current innovative activities in the applications of computers. In contrast to other journals, this journal focuses on emerging computer technologies with emphasis on the applicability to real world problems. Current areas of particular interest include, but are not limited to: architecture, networks, intelligent systems, parallel and distributed computing, software and information engineering, and computer applications (e.g., engineering, medicine, business, education, etc.). All papers are subject to peer review before selection.

A. Procedure for Submission of a Technical Paper for Consideration

1. Email your manuscript to the Editor-in-Chief, Dr. Fred Harris, Jr., Fred.Harris@cse.unr.edu.

2. Illustrations should be high quality (originals unnecessary).

3. Enclose a separate page (or include in the email message) the preferred author and address for correspondence. Also, please include email, telephone, and fax information should further contact be needed.

B. Manuscript Style:

1. The text should be **double-spaced** (12 point or larger), **single column** and **single-sided** on 8.5 X 11 inch pages.
2. An informative abstract of 100-250 words should be provided.
3. At least 5 keywords following the abstract describing the paper topics.
4. References (alphabetized by first author) should appear at the end of the paper, as follows: author(s), first initials followed by last name, title in quotation marks, periodical, volume, inclusive page numbers, month and year.
5. Figures should be captioned and referenced.

C. Submission of Accepted Manuscripts

1. The final complete paper (with abstract, figures, tables, and keywords) satisfying Section B above in **MS Word format** should be submitted to the Editor-in-Chief.
2. The submission may be on a CD/DVD or as an email attachment(s) . **The following electronic files should be included:**

    - Paper text (required).
    - Bios (required for each author). Integrate at the end of the paper.
    - Author Photos (jpeg files are required by the printer, these also can be integrated into your paper).
    - Figures, Tables, Illustrations. These may be integrated into the paper text file or provided separately (jpeg, MS Word, PowerPoint, eps).

3. Specify on the CD/DVD label or in the email the word processor and version used, along with the title of the paper.

4. Authors are asked to sign an ISCA copyright form (http://www.isca-hq.org/j-copyright.htm), indicating that they are transferring the copyright to ISCA or declaring the work to be government-sponsored work in the public domain. Also, letters of permission for inclusion of non-original materials are required.

Publication Charges

After a manuscript has been accepted for publication, the contact author will be invoiced for publication charges of **$50.00 USD** per page (in the final IJCA two-column format) to cover part of the cost of publication. For ISCA members, $100 of publication charges will be waived if requested.

January 2014