Computing and Informatics, Vol. 39, 2020, 881–906, doi: 10.31577/cai_2020_5_881

OPTIMAL FEATURE SUBSET SELECTION BASED ON COMBINING DOCUMENT FREQUENCY AND TERM FREQUENCY FOR TEXT CLASSIFICATION

Thirumoorthy Karpagalingam, Muneeswaran Karuppaiah

Department of Computer Science and Engineering
Mepco Schlenk Engineering College, Sivakai
Tamilnadu, India
e-mail: {kthirumoorthy, kmuni}@mepcoeng.ac.in

Abstract. Feature selection plays a vital role to reduce the high dimension of the feature space in the text document classification problem. The dimension reduction of feature space reduces the computation cost and improves the text classification system accuracy. Hence, the identification of a proper subset of the significant features of the text corpus is needed to classify the data in less computational time with higher accuracy. In this proposed research, a novel feature selection method which combines the document frequency and the term frequency (FS-DFTF) is used to measure the significance of a term. The optimal feature subset which is selected by our proposed work is evaluated using Naive Bayes and Support Vector Machine classifier with various popular benchmark text corpus datasets. The experimental outcome confirms that the proposed method has a better classification accuracy when compared with other feature selection techniques.

Keywords: Feature selection, text classification, document frequency, term frequency

1 INTRODUCTION

Nowadays millions of million users contribute a huge amount of information in the form of unstructured text data: movie/product reviews and feedbacks, social media tweets, and personal blogs are stored in the WWW repository. The organization of those unstructured text documents is a challengeable task. Text classification is

used to organize those documents in a proper way and to extract the information from that unstructured text corpus. The text classification is a supervised learning algorithm, which uses the training data associated with class labels to assign the text document to the appropriate categories [25, 32]. Text classification is used in topic detection [4], spam e-mail filtering [14, 8], e-mail classification [10, 41], author identification [7, 34] and web page classification [2, 6]. The atomic element or indivisible unit of the text document set is called a feature or word or term. The text classification system uses the vector space model to represent the text. The text corpus D is a set of unstructured text documents and is denoted as $D = \{d_1, d_2, d_3, \ldots, d_n\}$. Features are extracted from the text corpus and are denoted as $T = [t_1, t_2, t_3, \ldots, t_m]$. Each document d_i, $1 < i < n$ is represented as a feature vector $\langle w_{i1}, w_{i2}, w_{i3}, \ldots, w_{im}\rangle$ where w_{ij} denotes the frequency of feature t_j appearing in document d_i [1, 23]. The typical Document Term Matrix (DTM) is generally of sparse in nature and is shown in Table 1.

Doc	t_1	t_2	...	t_i	...	t_m
d_1	w_{11}	w_{12}		w_{1i}		w_{1m}
d_2	w_{21}	w_{22}		w_{2i}		w_{2m}
$\vdots$						
d_n	w_{n1}	w_{n2}		w_{ni}		w_{nm}

Table 1. Document term matrix

The high dimension of the feature space may contain the uninformative/irrelevant features (noise features), which reduce the accuracy of the classification system [40]. The uninformative feature (noise feature) has no information about the category. For example, if a word appears in all the text documents in the text corpus, that word is not at all useful to predict the class label. In order to reduce the dimension of the feature space as well as to improve the accuracy in text classification problems, feature selection plays a vital role [5, 11, 12, 13, 17, 9, 16]. Let F be the feature set having 'f' number of features, then we can coin the $2^f - 1$ (except empty set) number of different subsets of features. If we work with all the subsets, it would increase the computing cost. Feature selection is a process to select the optimal best feature subset space from the original feature space [33]. There are two types of feature selection methods:

1. Filter-based and
2. Wrapper-based.

Filter based feature selection method uses the various scoring methodologies to assign the importance score to each feature, and top-N features are selected based on the relevance score. Filter based methods are independent of classification models. Computationally the filter methods are faster. The wrapper methods [15, 3, 39] are based on attribute subset selection. The wrapper-based methods depend on the

classification model and search algorithms. The hybrid feature selection [12, 36] method uses both filter-based and wrapper-based method.

The rest of the paper is organized as follows: Section 2 briefly describes various feature selection approaches. Section 3 focuses on the proposed feature selection method. The classifiers used in the experiments are discussed in Section 4. The experimental results from the various datasets are discussed in Section 5 followed by the concluding remarks in Section 6.

2 RELATED WORK

The primary goals of the feature selection methods are to select the most appropriate features and ignore the irrelevant and redundant features [31, 21, 22]. Many feature selection algorithms are proposed for feature selection in text classification [18, 19]. In this section, we will discuss the existing feature selection methods, including Document Frequency (DF), Balanced Accuracy (ACC2), Distinguishing Feature Selector (DFS), Normalized Difference Measure (NDM), and Mutual Information (MI), which all are based on document frequency, and Information Gain (IG) is based on entropy methods. The document frequency in the collection of various scenarios is described in the contingency Table 2. Let C_k and $\overline{C_k}$ be two different categories: k (positive class) and other than "k" categories (negative class). Also, let the term t_i and $\overline{t_i}$ represent the presence and absence of the given term, respectively.

Terms/Category	t_i (Presence of the Term)	$\overline{t_i}$ (Absence of the Term)
C_K (belongs to)	tp	fn
$\overline{C_K}$ (does not belong to)	fp	tn

Table 2. Contingency table

tp: true positive count denotes the number of documents, which contain the term t_i and those documents belong to the category C_k (positive class)

fn: false negative count denotes the number of documents, which do not contain the term t_i and those documents belong to the category C_k (positive class)

fp: false positive count denotes the number of documents, which contain the term t_i and those documents do not belong to the category C_k (negative class)

tn: true negative count denotes the number of documents, which do not contain the term t_i and those documents do not belong to the category C_k (negative class)

The brief summary of preliminary notations, which are used in this work, is shown in Table 3.

Notation	Values	Description
K		number of Category in the dataset
N	$\mathrm{tp}+\mathrm{fn}+\mathrm{fp}+\mathrm{tn}$	number of documents in dataset
N_i	$\mathrm{tp}+\mathrm{fp}$	number of documents containing term t_i
N_k	$\mathrm{tp}+\mathrm{fn}$	number of documents belonging to category C_k
N_{ik}	tp	number of documents containing term t_i and belonging to C_k
$P(t_i)$	$\frac{N_i}{N}$	probabilities of presence of the term t_i
$\mathrm{P}(\overline{t_i})$	$\frac{(N-N_i)}{N}$	probabilities of absence of the term t_i
$P(C_k)$	$\frac{N_k}{N}$	probability of class C_k
$P(C_k\|t_i)$	$\frac{N_{ik}}{N_i}$	conditional probabilities of class C_k given presence of term t_i
$P(C_k\|\overline{t_i})$	$\frac{fn}{(N-N_i)}$	conditional probabilities of class C_k given absence of term t_i
$P(t_i\|C_k)$	$\frac{N_{ik}}{N_k}$	conditional probabilities of the presence of term t_i given class C_k
$P(t_i\|\overline{C_K})$	$\frac{fp}{(N-N_k)}$	conditional probability of the presence of term t_i given class other than C_k
$P(\overline{t_i}\|C_k)$	$\frac{fn}{N_k}$	conditional probability of absence of term t_i given class C_k
tf_{ij}		term frequency (occurrence) of term t_i in the document d_j
tf_i	$\sum_{j=1}^{N} tf_{ij}$	term frequency (occurrence) of term t_i in the entire dataset
tf_{ik}		term frequency (occurrence) of term t_i in the Category C_k
μ_i	$\frac{tf_i}{N}$	mean term frequency of term t_i in the entire dataset
μ_{ik}	$\frac{tf_{ik}}{N_k}$	mean term frequency of term t_i in the Category C_k
σ_i	$\sqrt{\frac{\sum_{j=1}^{N}(tf_{ij}-\mu_i)^2}{N}}$	standard deviation of term frequency of term t_i in the entire dataset
σ_{ik}	see Equation (13)	standard deviation of term frequency of term t_i in the Category C_k

Table 3. Preliminary notation

2.1 Document Frequency (DF)

Document Frequency (DF) [26, 38, 28] of the term 't' is the simplest feature selection method and is based on the number of documents containing term 't'. The DF method considers that the rare frequency terms are non-informative for text categorization. It ignores the impact on category information. This method considers the impact on the positive class and ignores the negative class. The document frequency

of the term 't' is calculated as follows:

$$\mathrm{DF}(t) = \mathrm{tp} + \mathrm{fp}. \tag{1}$$

2.2 Balanced Accuracy (ACC2)

Accuracy is one of the feature ranking metric, which considers the difference between true positives and false positives of a term, and it supports strong positive features. Balanced Accuracy (ACC2) [11, 30] is a variant of accuracy which ranks the features based on the absolute difference of the true positive rate (tpr) and false positive rate (fpr). The tpr and fpr of the term 't' are calculated as follows:

$$\mathrm{tpr}(t) = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fn}}, \tag{2}$$

$$\mathrm{fpr}(t) = \frac{\mathrm{fp}}{\mathrm{tn} + \mathrm{fp}}. \tag{3}$$

Balanced Accuracy ignores the influence of term frequency, which uses the absolute difference between tpr and fpr as follows:

$$\mathrm{ACC2}(t) = |\mathrm{tpr}(t) - \mathrm{fpr}(t)|. \tag{4}$$

2.3 Information Gain (IG)

Information Gain (IG) [28, 24] is an entropy-based evaluation technique, which is used in machine learning applications. It refers to the difference between the information entropy produced by the presence and absence of a term in the document. The expression for IG is as follows:

$$\begin{aligned} IG(t_i) = &- \sum_{k=1}^{K} P(C_k) \log P(C_k) \\ &+ P(t_i) \sum_{k=1}^{K} P(C_k|t_i) \log P(C_k|t_i) \\ &+ P(\overline{t_i}) \sum_{k=1}^{K} P(C_k|\overline{t_i}) \log P(C_k|\overline{t_i}). \end{aligned} \tag{5}$$

2.4 Mutual Information (MI)

Mutual Information (MI) [28, 27] is a measure between two random variables, which quantifies the amount of information gained about one variable through another

variable. The computation of mutual information for a given term t_i is given as

$$MI(t_i, C_k) = \log_2 \frac{P(t_i|C_k)}{P(t_i)}, \tag{6}$$

$$MI(t_i) = \sum_{k=1}^{K} P(C_k) * MI(t_i, C_k). \tag{7}$$

2.5 Distinguishing Feature Selector

Uysal and Gunal [37] proposed the probabilistic based feature selection scheme as a Distinguishing Feature Selector (DFS). DFS assigns the high score to the term, which frequently occurs in one of the categories and does not occur in the other categories, is distinctive; A term which frequently occurs in all the class is irrelevant; it must be assigned with a low score. DFS ignores the significance of term frequency information in their scoring mechanism. DFS can be formulated

$$\mathrm{DFS}(t_i) = \sum_{k=1}^{K} \frac{P(C_k|t_i)}{1 + P(\overline{t_i}|C_k) + P(t_i|\overline{C_k})}. \tag{8}$$

DFS score of the feature lies between 0.5 and 1.0. The most discriminating terms have an importance score that is close to 1.0 and the least discriminating terms are assigned with the significance score is 0.5.

2.6 Normalized Difference Measure (NDM)

NDM algorithm [30] uses the ACC2 value divided by the minimum of tpr and fpr to indicate the importance of the feature. The value of NDM is estimated as follows:

$$\mathrm{NDM}(t) = \frac{|\mathrm{tpr}(t) - \mathrm{fpr}(t)|}{\min(\mathrm{tpr}(t), \mathrm{fpr}(t))}. \tag{9}$$

If $\min(\mathrm{tpr}, \mathrm{fpr}) = 0$, then to avoid divide by zero exception, it is replaced by small value. NDM method ignores the influence of term frequency. In addition, NDM equally ranks the terms having equal weight value regardless of the value of $|\mathrm{tpr} - \mathrm{fpr}|$.

All of the above-mentioned feature selection schemes ignore the significance of the term frequency. To illustrate the significance of term frequency, a sample data is shown in 4, which consists of 10 documents, grouped under 3 categories {C1, C2, C3}. There are seven distinct terms from the 10 documents and they are: {lion, tiger, bear, goat, deer, horse, panda}. The following handwork experiments on the sample dataset shows the significance of the term frequency.

Table 4 shows the Document Term Matrix (DTM) of the sample dataset.

Table 5 shows the importance score of above mentioned selection scheme.

List of issues for ignoring the term frequency:

Documents	**Category**	**lion**	**bear**	**tiger**	**goat**	**deer**	**horse**	**panda**
d_1	C_1	1	10	10	3	15	30	25
d_2	C_1	1	10	10	5	15	20	10
d_3	C_1	1	10	10	4	0	40	2
d_4	C_2	1	10	1	10	0	0	0
d_5	C_2	1	10	1	10	0	0	0
d_6	C_2	1	10	1	10	20	0	0
d_7	C_2	1	10	1	10	20	0	0
d_8	C_3	1	10	1	1	3	0	0
d_9	C_3	1	10	1	1	0	0	0
d_{10}	C_3	1	10	1	1	0	0	0

Table 4. Document term matrix of the sample dataset

Documents	**lion**	**bear**	**tiger**	**goat**	**deer**	**horse**	**panda**
DF	10	10	10	10	5	3	3
ACC2	0	0	0	0	0.142	0.628	0.628
IG	0	0	0	0	0.049	0.881	0.881
MI	0	0	0	0	−0.05	0.52	0.52
DFS	0.5	0.5	0.5	0.5	0.516	1.0	1.0
NDM	0	0	0	0	0.38	6.286	6.286

Table 5. Importance scores of the term in sample dataset

- The terms 'lion', 'tiger', 'bear', 'goat' appeared in all the documents. The existing feature selection method ACC2, NDM, IG and MI consider these terms are useless. And DFS says that all the four (lion, tiger, bear, goat) are the same. While comparing lion and tiger, tiger may contribute more to the category C1 also goat may contribute more to category C2 based on term frequency. So we cannot provide the equal weightage to these terms.
- The terms 'horse' and 'panda' only appeared in the category C1. ACC2, NDM, IG, DFS provide equal weightage. Based on the term frequency horse must have highest significance score.

In order to address these problems, we combine the document frequency and term frequency information to select the optimal feature subset.

3 PROPOSED WORK

The feature selection method assigns high significant scores to the more informative features and lower significant scores to less informative/irrelevant features. The above mentioned feature selection scheme ranks the feature based on how the term contributes to the categorization based on document frequency. We propose the new feature selection scheme which integrates the document frequency contribution and term frequency contribution. The proposed feature selection method FS-DFTF

assigns the significance score based on the following:

- FS-DFTF assigns a high significance score to a term which frequently occurs in a single category and does not occur in the other category, it is an informative term.
- FS-DFTF assigns a low significance score to a term which frequently occurs in all the categories, it is irrelevant feature.

We consider the term frequency distribution in two levels: i) the frequency of the term between the category level, and ii) the frequency of the term within the category level. The system design of our proposed work is depicted in Figure 1.

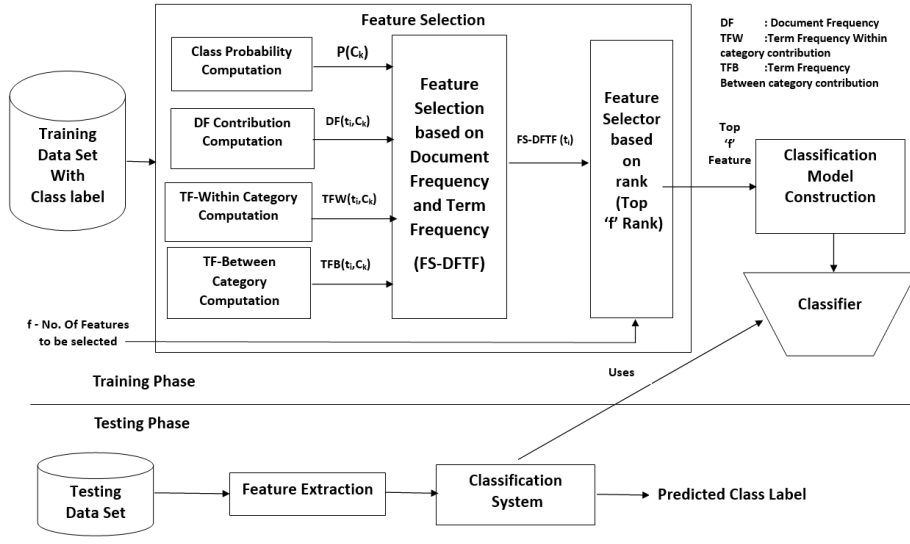


Figure 1. The architectural framework for text classification system employing FS-DFTF

In order to integrate the term frequency information, we are using the standard deviation of term frequency and mean of term frequency at both levels (between category level and within category level). The most informative feature must have high mean frequency and less standard deviation of term frequency. Based on this, we formulate the FS-DFTF as follows:

$$\text{FS} - \text{DFTF}(t_i) = \sum_{k=1}^{K} (P(C_k) * \theta(t_i, C_k) * \Phi(t_i, C_k) * \Psi(t_i, C_k)) \tag{10}$$

where $\theta(t_i, C_k)$ indicates the Document Frequency contribution of the term t_i in category C_k, $\Phi(t_i, C_k)$ indicates term frequency contribution between the category level, $\Psi(t_i, C_k)$ indicates term frequency contribution within the category level.

Document frequency contribution can be computed as follows:

$$\theta(t_i, C_k) = \frac{P(C_k|t_i) - P(C_k|\overline{t_i})}{1 + P(\overline{t_i}|C_k) + P(t_i|\overline{C_k})}. \tag{11}$$

In this work, we consider $P(C_k|\overline{t_i}) = 0$ if $P(\overline{t_i}) = 0$, to avoid division by zero errors. The computed document frequency contribution $\theta(t_i, C_k)$ over the sample dataset is shown in Table 6.

	$P(C_k\|t_i)$			$P(C_k\|\overline{t_i})$			$P(\overline{t_i}\|C_k)$			$P(t_i\|\overline{C_k})$			$\theta(t_i, C_k)$		
Term	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3	C_1	C_2	C_3
lion	0.3	0.4	0.3	0	0	0	0	0	0	1	1	1	0.15	0.2	0.15
bear	0.3	0.4	0.3	0	0	0	0	0	0	1	1	1	0.15	0.2	0.15
tiger	0.3	0.4	0.3	0	0	0	0	0	0	1	1	1	0.15	0.2	0.15
goat	0.3	0.4	0.3	0	0	0	0	0	0	1	1	1	0.15	0.2	0.15
deer	0.399	0.4	0.199	0.199	0.4	0.399	0.333	0.5	0.666	0.429	0.5	0.571	0.114	0	−0.089
horse	1	0	0	0	0.571	0.429	0	1	1	0	0.5	0.429	1	−0.229	−0.176
panda	1	0	0	0	0.571	0.429	0	1	1	0	0.5	0.429	1	−0.229	−0.176

Table 6. Document frequency contribution $\theta(t_i, C_k)$ over the sample dataset

Term frequency contribution between the category levels can be computed as follows:

$$\Phi(t_i, C_k) = \begin{cases} \frac{N_{ik}}{N_i} * \frac{tf_{ik}}{tf_i}, & \sigma_i > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

If the standard deviation term frequency of the term t_i is zero ($\sigma_i = 0$) then the term t_i appears in all the document with equal term frequency. For the classification task, that term t_i is not useful. Hence, we can say that $\Phi(t_i, C_k) = 0$. The computed term frequency contribution between the category levels $\Phi(t_i, C_k)$ over the sample dataset is shown in Table 7.

	N_i	N_{ik}			$\mathbf{tf}_i$	$\mathbf{tf}_{ik}$			σ_i	$\Phi(t_i, C_k)$		
Term		C_1	C_2	C_3		C_1	C_2	C_3		C_1	C_2	C_3
lion	10	3	4	3	10	3	4	3	0	0	0	0
bear	10	3	4	3	100	30	40	30	0	0	0	0
tiger	10	3	4	3	37	30	4	3	4.35	0.243	0.043	0.024
goat	10	3	4	3	55	12	40	3	4.09	0.065	0.290	0.016
deer	5	2	2	1	73	30	40	3	8.98	0.164	0.219	0.008
horse	3	3	0	0	90	90	0	0	15.24	1.0	0	0
panda	3	3	0	0	37	37	0	0	8.11	1.0	0	0

Table 7. Term frequency contribution between the category level $\Phi(t_i, C_k)$ over the sample dataset

The Term Frequency contribution within the category level can be computed as follows:

$$\Psi(t_i, C_k) = \frac{1}{\sigma_{ik}} * \left[\frac{\frac{N_{ik}}{N_k} * \mu_{ik}}{\max_{\{j=1,2,\ldots,m\}} \left\{ \frac{N_{jk}}{N_k} * \mu_{jk} \right\}} \right] \tag{13}$$

where $\sigma_{ik} = \sqrt{\frac{\sum_{j=1}^{N} (tf_{ij} - \mu_{ik})^2 * I_{jk}}{N_k}}$, $I_{jk} = \begin{cases} 1, & d_j \in C_k \\ 0, & \text{otherwise} \end{cases}$, m is the number of distinct terms present in the Category C_k. This part represents, how much the term frequency of term t_i contributes to the category C_k while comparing other terms within a category. If σ_{ik} is zero then the term t_i appears in all the document of category C_k with equal term frequency. In that case, we may ignore the σ_{ik} component of feature t_i. So,

$$\Psi(t_i, C_k) = \frac{\frac{N_{ik}}{N_k} * \mu_{ik}}{\max_{\{j=1,2,\ldots,m\}} \left\{ \frac{N_{jk}}{N_k} * \mu_{jk} \right\}}, \text{ if } \sigma_{ik} = 0. \tag{14}$$

If the standard deviation of term frequency of the term is zero ($\sigma_i = 0$) then the term t_i present in all the document with equal term frequency. That term is a not useful for classification task. So we can say that $\Phi(t_i, C_k) = 0$ and $\Psi(t_i, C_k) = 0$. The computed term frequency contribution within the category levels $\Psi(t_i, C_k)$ over the sample dataset is shown in Table 8.

	N_{ik}			μ_{ik}			σ_{ik}			$\Psi(t_i, C_k)$		
Term	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**	**C1**	**C2**	**C3**
lion	3	4	3	1	1	1	0	0	0	0	0	0
bear	3	4	3	10	10	10	0	0	0	0	0	0
tiger	3	4	3	10	1	1	0	0	0	0.333	0.1	0.1
goat	3	4	3	4	10	1	1	0	0	0.13	1	0.1
deer	2	2	1	10	10	1	8.660	11.547	1.732	0.222	0.5	0.033
horse	3	0	0	30	0	0	10	0	0	1	0	0
panda	3	0	0	12.333	0	0	11.676	0	0	0.411	0	0

Table 8. Term frequency contribution within the category level $\Psi(t_i, C_k)$ over the sample dataset

Finally, the FS-DFTF score of given sample datadet is shown in Table 9.

Term	**lion**	**bear**	**tiger**	**goat**	**deer**	**horse**	**panda**
FS-DFTF	0	0	0.004	0.023	0.001	0.3	0.123
FS-DFTF Rank	6	6	4	3	5	1	2

Table 9. FS-DFTF score of the sample dataset

In order to show the working principles of FS-DFTF, the sample dataset and related results are shown in the above mentioned Tables 4, 5, 6, 7, 8, 9. The real

performance of FS-DFTF on the popular benchmark datasets are presented briefly in the experimental work section.

3.1 The Pseudo Procedure for the FS-DFTF

Algorithm 1 describes the process of proposed work.

Algorithm 1: feature selection using FS-DFTF

Input:
D: Dataset with class label
f : Number of features to be selected
Output: F_{best}: selected f features

```
1  D ← textPreProcessing(D)
2  T : [t_1, t_2, t_3, ..., t_m] ← Tokenizer(D) // m-numbers of unique features in D
3  for each category C_k in D
4     P(C_k) = computeClassProbability(D)
5  end
6  for each term t_i in T
7     for each category C_k in D
8        θ(t_i, C_k) = computeDFContribution(D) // use eqn.(10)
9        Φ(t_i, C_k) = computeTFContributionBetweenCategoryLevel(D) // use
       eqn.(11)
10       Ψ(t_i, C_k) = computeTFContributionWithinCategoryLevel(D) // use
       eqn.(12)
11    end
12    FS_DFTF(t_i) = computeFSDFTF() //use eqn.(13)
13 end
14 F_best = SortAndSelect(FS_DFTF, f)
15 return   F_best
```

Line 1 shows the preprocessing. The preprocessing steps are lower casings, removing punctuations, numbers, stop words and finally stemming. Line 2 tokenize the entire dataset to find the unique feature list. Lines 3–13 compute the FS-DFTF score of each term t_i in the feature list. Line 14 sorts the feature based on FS-DFTF importance score value and selects the top 'f' feature.

4 DATASET AND EXPERIMENTAL SETUP

In this section, we briefly discuss the datasets, classification algorithm and the evaluation metric to evaluate the performance of the proposed feature selection measure for text classification.

4.1 Dataset

In this work, we have experimented with four distinct datasets (WebKB, SMS, BBC News and 10Newsgroups) used for the assessment of our proposed feature selection method [dataset[1]]. WebKB is a collection of web pages collected by the World Wide Knowledge Base. These pages were collected from computer science departments of various universities in 1997. The web pages are classified as various classes: student, faculty, staff, department, course, project, and other. For the experimental works we chose the class label (project, course, faculty, student) documents. Table 10 describes the properties of WebKB dataset. SMS Dataset is a collection of SMS, which is labeled as Spam or Ham. It contains 5 574 labeled SMS message. Description of the SMS dataset is shown in Table 11. BBC Dataset contains 2 225 text documents from the BBC news website, which are classified into five categories (business, entertainment, politics, sport, tech), Table 12 shows the document distribution of BBC dataset. Newsgroups Dataset: The documents of Newsgroups dataset contains approximately 15 000 news documents, which are manually classified into 20 groups. In this work we have experimented with ten categories. News document distribution among the selected categories is shown in Table 13.

S. No	Category	Training Docs	Testing Docs	Total Docs
1	project	336	168	504
2	course	620	310	930
3	faculty	750	374	1 124
4	student	1 097	544	1 641
	Total	2 803	1 396	4 199

Table 10. WebKB dataset

S. No	Category	Training Docs	Testing Docs	Total Docs
1	Spam	436	311	747
2	Ham	2 838	1 989	4 827
	Total	3 274	2 300	5 574

Table 11. SMS dataset

4.2 Classifier Used

In this proposed research, we have used the Naive Bayes (NB) and Support Vector Machine classifiers to classify the unstructured documents. The primary idea of NB classifier [5, 31, 35] is to use the joint probabilities of the terms and class labels of

[1] dataset: `http://ana.cachopo.org/datasets-for-single-label-text-categorization`

S. No	Category	Training Docs	Testing Docs	Total Docs
1	Business	281	229	510
2	Entertainment	223	163	386
3	Politics	219	198	417
4	Sport	292	219	511
5	Tech	210	191	401
	Total	1 225	1 000	2 225

Table 12. BBC dataset

S. No	Category	Training Docs	Testing Docs	Total Docs
1	rec. autos	594	395	989
2	rec. motorcycles	598	398	996
3	rec. sport. baseball	597	397	994
4	rec. sport. hockey	600	399	999
5	sci. crypt	595	396	991
6	sci. electronics	591	393	984
7	sci. med	594	396	990
8	sci. space	593	394	987
9	soc. religion. christian	598	398	996
10	talk. politics. guns	545	364	909
	Total	5 905	3 930	9 835

Table 13. 10Newsgroup dataset

the training set to determine the class label of a given unknown document. Given the document d_j, the probability with each category C_k is computed as follows:

$$P(C_k|d_j) = \frac{P(d_j|C_k)}{P(d_j)} * P(C_k). \tag{15}$$

The class label for the document d_j, can be evaluated by,

$$\text{Label}(d_j) = \max_{k=1,2,\ldots,K} \{P(C_k|d_j)\}. \tag{16}$$

Support vector machine (SVM) is one of the best supervised learning algorithm which is used for classification and regression [10, 18, 19, 22, 29, 20]. SVM finds a hyper-plane or set of hyper-planes to separate the classes in the high dimensional space. The main objective of SVM is to find the decision boundary that is maximally away from any data point. SVM classifier finds the maximum margin hyper plane, which separate the two classes and the border of hyper-plane is defined by support vectors.

4.3 Evaluation Metric

There are four standard evaluation measures namely Accuracy, Precision, Recall, F-Score that are used to evaluate the proposed feature selection model for classifying the unstructured documents [28]. True positives (TP) are instances when the actual category of the tuple was positive and the predicted is also positive. True negatives (TN) are instances when the actual category of the tuple was negative and predicted is also negative. False positives (FP) are instances when the actual category of the tuple was negative and predicted is positive. False negatives (FN) are instances when the actual category of the tuple was positive and predicted as negative. The accuracy of a classification model is calculated as how much percentage of testing dataset documents are accurately classified.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}. \tag{17}$$

The Precision (P) is a measure of exactness. It refers what percentage of tuples classified as positive are actually positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{18}$$

The Recall (R) is a measure of completeness. It refers what percentage of positive tuples are classified as positive are actually positive.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{19}$$

The F-Score (F) is defined as the weighted harmonic mean of the Precision and Recall.

$$F = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{20}$$

5 RESULTS

In this section, a deep analysis is carried out to compare the FS-DFTF feature selection scheme against various filter based feature selection methods (DF, ACC2, IG, MI, DFS, and NDM) in terms of classifier accuracy, precision, recall and F-score. We have taken the measurement of experiments using a computer equipped with 2.30 GHz, Intel Core i5 processor and 8 GB RAM memory. The experiments are conducted with features of varying sizes such as 10, 50, 100, 200, 300, 500 and 1 000. We have used Python 3.7.3 for programming and matplotlib library to plot the performance graph. The performance of the proposed work FS-DFTF is shown in Figures 2, 3, 4, 5 and Tables 14, 15, 16, 17, 18, 19, 20, 21 on the above mentioned dataset respectively. In all the graphs, the X-axis represents the number of selected features and the Y-axis represents the corresponding classifier performance

in terms of accuracy. In most experimental results, the proposed method shows better accuracy than other contrast ones.

5.1 Performance Comparisons on the WebKB Dataset

The accuracy of the NB classifier and SVM classifier on the WebKB dataset are shown in Figures 2 a) and 2 b), respectively. According to Figure 2, the proposed FS-DFTF method surpasses the individual performance of all other methods in terms of accuracy using Naive Bayes classifier and SVM classifier. For the WebKB dataset, the optimum feature size is 200 with an accuracy of 89.18 % for NB classifier and 87.97 % for SVM classifiers. We observe that the accuracy curve of FS-DFTF is higher than that of other methods for both Naive Bayes and linear SVM classifiers.

Table 14 shows the Precision, Recall and F-Score of Naive Bayes classifier using FS-DFTF, MI, DFS, DF, ACC2, IG and NDM on the Webkb dataset when top 200 features are selected in feature space. The results show that FS-DFTF method has a higher number of instances correctly classified (1 245 instances over 1 396) than the six existing techniques and it improves the classification performance.

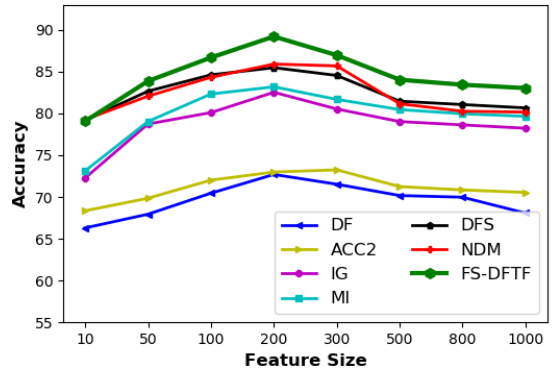
Algorithm	**Precision**	**Recall**	**F_1 Score**	**Accuracy in % (Correctly Classified Docs)**	**Error Rate in % (Incorrectly Classified Docs)**
DF	0.74	0.73	0.73	72.7 % (1 015)	27.3 % (381)
ACC2	0.74	0.73	0.73	72.99 % (1 019)	27.01 % (377)
IG	0.83	0.83	0.83	82.52 % (1 152)	17.48 % (244)
MI	0.84	0.83	0.83	83.17 % (1 161)	16.83 % (235)
DFS	0.86	0.85	0.86	85.46 % (1 193)	14.54 % (203)
NDM	0.86	0.86	0.86	85.89 % (1 199)	14.11 % (197)
FS-DETF	0.89	0.89	0.89	89.18 % (1 245)	10.82 % (151)

Table 14. Performance of FS-DFTF on WebKB dataset using NB classifier

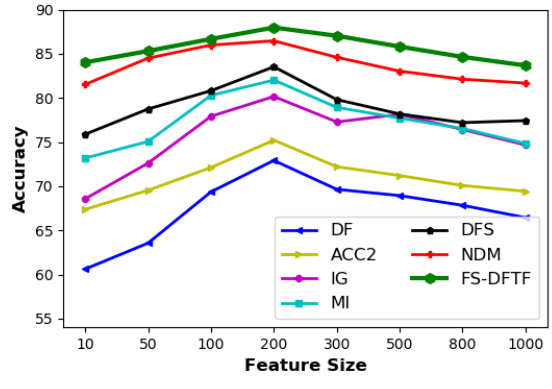
Table 15 shows the Precision, Recall and F-Score of SVM classifier using FS-DFTF, MI, DFS, DF, ACC2, IG and NDM on the Webkb dataset when top 200 features are selected in feature space. The results show that FS-DFTF method has a higher number of instances correctly classified (1 228 instances over 1 396) than the six existing techniques and it improves the classification performance.

5.2 Performance Comparisons on the SMS Dataset

Figure 3 shows the experimental results of text classification on the SMS dataset using NB and SVM classifiers. The curves in the figures indicate the various feature selection scheme. It can be seen from Figure 3 a) that the performance of FS-DFTF using the NB classifier is better than all other feature selection scheme. Also, Figure 3 b) shows that the performance of the proposed work using the SVM classifier has the highest accuracy while comparing other feature selection scheme. For the



a) NB classifier



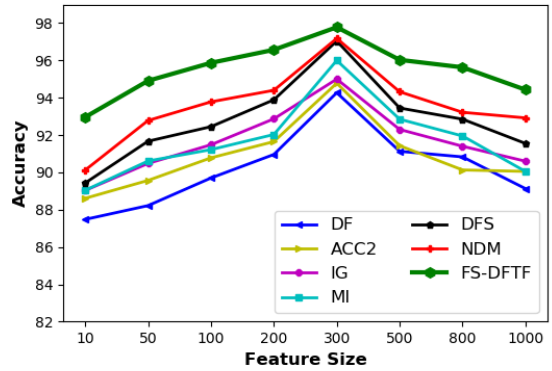
b) SVM classifier

Figure 2. Accuracy comparison for WebKb dataset using a) NB classifier b) SVM classifier

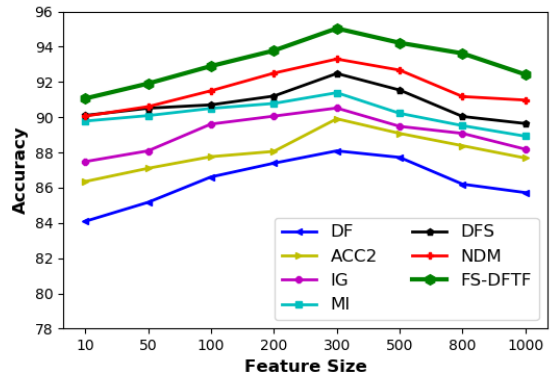
Algorithm	**Precision**	**Recall**	**F_1 Score**	**Accuracy in % (Correctly Classified Docs)**	**Error Rate in % (Incorrectly Classified Docs)**
DF	0.74	0.73	0.73	72.92 % (1 018)	27.08 % (378)
ACC2	0.77	0.75	0.76	75.21 % (1 050)	24.79 % (346)
IG	0.81	0.8	0.8	80.16 % (1 119)	19.84 % (277)
MI	0.83	0.82	0.82	82.02 % (1 145)	17.98 % (251)
DFS	0.84	0.84	0.84	83.52 % (1 166)	16.48 % (194)
NDM	0.87	0.86	0.87	86.46 % (1 207)	13.54 % (189)
FS-DETF	0.88	0.88	0.88	87.97 % (1 228)	12.03 % (168)

Table 15. Performance of FS-DFTF on WebKB dataset using SVM classifier

SMS dataset, the optimum feature size is 300 with an accuracy of 97.78 % for NB classifier and 95.04 % for SVM classifiers.



a) NB classifier



b) SVM classifier

Figure 3. Accuracy comparison for SMS dataset using a) NB classifier b) SVM classifier

While selecting top 300 features in feature space, the Precision, Recall and F-Score of Naive Bayes classifier using FS-DFTF, MI, DFS, DF, ACC2, IG and NDM on the Webkb dataset is shown in Table 16. The results show that FS-DFTF method has a higher number of instances correctly classified (2 249 instances over 2 300) than the six existing techniques and it improves the classification performance.

Table 17 shows the Precision, Recall and F-Score of SVM classifier using FS-DFTF, MI, DFS, DF, ACC2, IG and NDM on the SMS dataset when top 300 features are selected in feature space. The results show that FS-DFTF method has

Algorithm	Precision	Recall	F_1 Score	Accuracy in % (Correctly Classified Docs)	Error Rate in % (Incorrectly Classified Docs)
DF	0.95	0.94	0.95	94.27 % (2 168)	5.74 % (132)
ACC2	0.96	0.95	0.95	94.78 % (2 180)	5.21 % (120)
IG	0.96	0.95	0.95	95.0 % (2 185)	5.0 % (115)
MI	0.97	0.96	0.96	96.0 % (2 208)	4.0 % (92)
DFS	0.97	0.97	0.97	97.04 % (2 232)	2.96 % (68)
NDM	0.97	0.97	0.97	97.18 % (2 235)	2.82 % (65)
FS-DETF	0.98	0.98	0.98	97.78 % (2 249)	2.22 % (51)

Table 16. Performance of FS-DFTF on SMS dataset using NB classifier

a higher number of instances correctly classified (2 186 instances over 2 300) than the six existing techniques and it improves the classification performance.

Algorithm	Precision	Recall	F_1 Score	Accuracy in % (Correctly Classified Docs)	Error Rate in % (Incorrectly Classified Docs)
DF	0.92	0.88	0.89	88.09 % (2 026)	11.91 % (274)
ACC2	0.93	0.90	0.91	89.91 % (2 068)	10.09 % (232)
IG	0.93	0.91	0.91	90.52 % (2 082)	9.48 % (218)
MI	0.94	0.91	0.92	91.39 % (2 102)	8.61 % (198)
DFS	0.94	0.92	0.93	92.48 % (2 127)	7.52 % (173)
NDM	0.95	0.93	0.94	93.3 % (2 146)	6.7 % (154)
FS-DETF	0.96	0.95	0.95	95.04 % (2 186)	4.96 % (114)

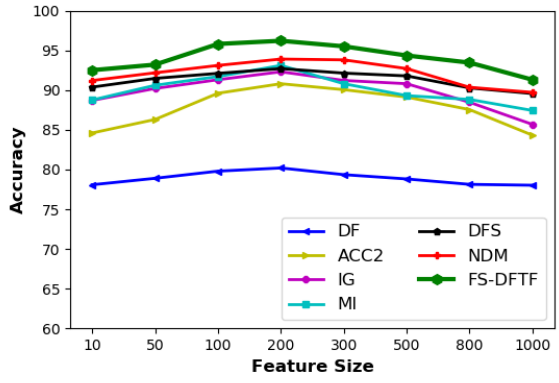
Table 17. Performance of FS-DFTF on SMS dataset using SVM classifier

5.3 Performance Comparisons on the BBC Dataset

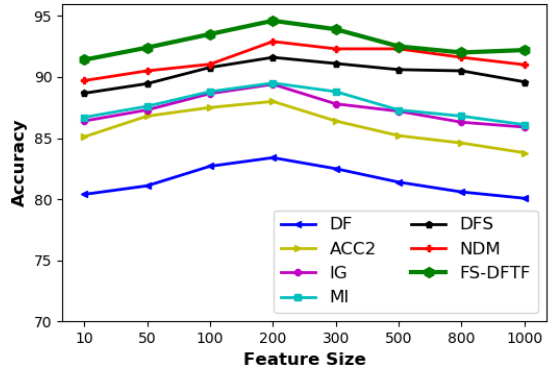
The accuracy of the NB classifier and SVM classifier on the BBC dataset are shown in Figures 4 a) and 4 b), respectively. According to Figure 4, the proposed FS-DFTF method surpasses the individual performance of all other methods in terms of accuracy using Naive Bayes classifier and SVM classifier. For the BBC dataset, the optimum feature size is 200 with an accuracy of 96.2 % for NB classifier and 94.6 % for SVM classifiers. We observe that the accuracy curve of FS-DFTF is higher than that of other methods for both Naive Bayes and linear SVM classifiers.

Table 18 shows the Precision, Recall and F-Score of Naive Bayes classifier using FS-DFTF, MI, DFS, DF, ACC2, IG and NDM on the BBC dataset when top 200 features are selected in feature space. The results show that FS-DFTF method has a higher number of instances correctly classified (962 instances over 1 000) than the six existing techniques and it improves the classification performance.

Table 19 shows the Precision, Recall and F-Score of SVM classifier using FS-DFTF, MI, DFS, DF, ACC2, IG and NDM on the BBC dataset when top 200 fea-



a) NB classifier



b) SVM classifier

Figure 4. Accuracy comparison for BBC dataset using a) NB classifier b) SVM classifier

Algorithm	**Precision**	**Recall**	**F_1 Score**	**Accuracy in % (Correctly Classified Docs)**	**Error Rate in % (Incorrectly Classified Docs)**
DF	0.8	0.8	0.8	80.2 % (802)	19.8 % (19.8)
ACC2	0.91	0.91	0.91	90.8 % (908)	9.2 % (92)
IG	0.92	0.92	0.92	92.3 % (92.3)	7.7 % (77)
MI	0.93	0.93	0.93	93.1 % (931)	6.9 % (69)
DFS	0.93	0.93	0.93	92.7 % (927)	7.3 % (73)
NDM	0.94	0.94	0.94	93.9 % (939)	6.1 % (61)
FS-DETF	0.96	0.96	0.96	96.2 % (962)	3.8 % (38)

Table 18. Performance of FS-DFTF on BBC News corpus using NB classifier

tures are selected in feature space. The results show that FS-DFTF method has a higher number of instances correctly classified (946 instances over 1 000) than the six existing techniques and it improves the classification performance.

Algorithm	Precision	Recall	F_1 Score	Accuracy in % (Correctly Classified Docs)	Error Rate in % (Incorrectly Classified Docs)
DF	0.83	0.83	0.83	83.4 % (834)	16.6 % (166)
ACC2	0.88	0.88	0.88	88.0 % (880)	12.0 % (120)
IG	0.89	0.89	0.89	89.4 % (894)	10.6 % (106)
MI	0.9	0.9	0.9	89.5 % (895)	10.5 % (105)
DFS	0.92	0.92	0.92	91.6 % (916)	8.4 % (84)
NDM	0.93	0.93	0.93	92.9 % (929)	7.1 % (71)
FS-DETF	0.95	0.95	0.95	94.6 % (946)	5.4 % (54)

Table 19. Performance of FS-DFTF on BBC News corpus using SVM classifier

5.4 Performance Comparisons on the 10Newsgroup Dataset

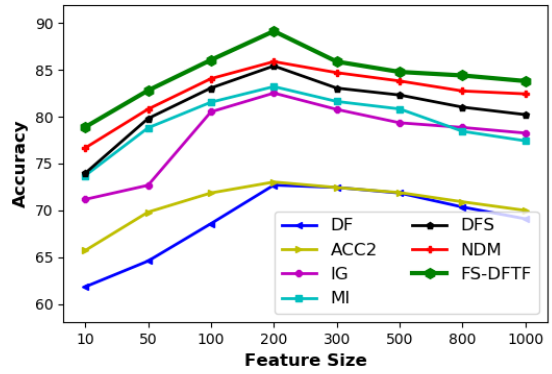
Figure 5 shows the experimental results of text classification on the 10Newsgroup dataset using NB and SVM classifiers. The curves in the figures indicate the various feature selection scheme. It can be seen from Figure 5 a) that the performance of FS-DFTF using the NB classifier is better than all other feature selection scheme. Also, Figure 5 b) shows that the performance of the proposed work using the SVM classifier has the highest accuracy while comparing other feature selection scheme. For the SMS dataset, the optimum feature size is 300 with an accuracy of 89.16 % for NB classifier and 86.77 % for SVM classifiers.

While selecting top 200 features in feature space, the Precision, Recall and F-Score of Naive Bayes classifier using FS-DFTF, MI, DFS, DF, ACC2, IG and NDM on the 10Newsgroup dataset is shown in Table 20. The results show that FS-DFTF method has a higher number of instances correctly classified (3 504 instances over 3 930) than the six existing techniques and it improves the classification performance.

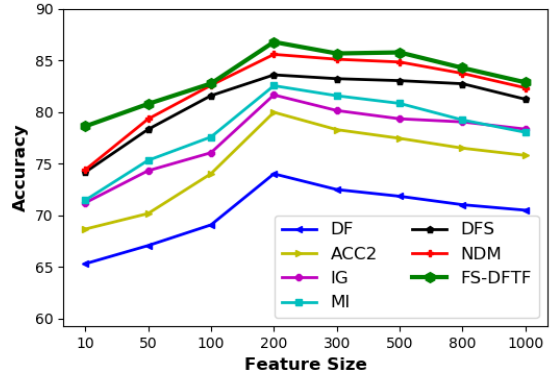
Table 21 shows the Precision, Recall and F-Score of SVM classifier using FS-DFTF, MI, DFS, DF, ACC2, IG and NDM on the 10Newsgroup dataset when top 200 features are selected in feature space. The results show that FS-DFTF method has a higher number of instances correctly classified (3 410 instances over 3 930) than the six existing techniques and it improves the classification performance.

6 VALIDITY THREATS

In this section, we discuss the validity threats for our proposed filter based feature selection scheme. We have identified two validity threats:



a) NB classifier



b) SVM classifier

Figure 5. Accuracy comparison for 10Newsgroup dataset using a) NB classifier b) SVM classifier

Algorithm	**Precision**	**Recall**	**F_1 Score**	**Accuracy in % (Correctly Classified Docs)**	**Error Rate in % (Incorrectly Classified Docs)**
DF	0.73	0.73	0.73	72.70 % (2 857)	27.30 % (1 073)
ACC2	0.73	0.73	0.73	73.03 % (2 870)	26.97 % (1 060)
IG	0.83	0.83	0.83	82.54 % (3 244)	17.46 % (686)
MI	0.83	0.83	0.83	83.23 % (3 271)	16.77 % (659)
DFS	0.85	0.85	0.85	85.44 % (3 358)	14.56 % (572)
NDM	0.86	0.86	0.86	85.90 % (3 376)	14.10 % (554)
FS-DETF	0.89	0.89	0.89	89.16 % (3 504)	10.84 % (426)

Table 20. Performance of FS-DFTF on 10Newsgroup dataset using NB classifier

Algorithm	Precision	Recall	F_1 Score	Accuracy in % (Correctly Classified Docs)	Error Rate in % (Incorrectly Classified Docs)
DF	0.74	0.74	0.74	74.02 % (2 909)	25.98 % (1 021)
ACC2	0.8	0.8	0.8	79.97 % (3 143)	20.03 % (787)
IG	0.82	0.82	0.82	81.65 % (3 209)	18.35 % (721)
MI	0.83	0.83	0.83	82.54 % (3 244)	17.46 % (686)
DFS	0.84	0.84	0.84	83.59 % (3 285)	16.41 % (645)
NDM	0.86	0.86	0.86	85.57 % (3 363)	14.43 % (567)
FS-DETF	0.87	0.87	0.87	86.77 % (3 410)	13.23 % (520)

Table 21. Performance of FS-DFTF on 10Newsgroup dataset using SVM classifier

Less or no contribution of term frequency (TF) to the text corpus. The Sarcasm headlines dataset[2] is a collection of sarcastic headlines. It contains more than 25 000 headlines. These headlines are classified into two categories (Sarcastic, Non sarcastic). In this text corpus, each document is a news headline which contains non repeated words. As a result, the term frequency (TF) does not contribute to assign the significance score to a term.

Computational cost. Even though, the performamce of the proposed FS-DFTF feature selection scheme outperformed the other feature selection scheme, the proposed method takes more computation time. Because, the proposed work uses both DF contribution and TF contribution to assign the significance score to each term which incurs some additional computational cost.

7 CONCLUSION

In this work, we propose a new filter based feature selection scheme which combines the document frequency and term frequency of the term. The performance of the proposed work FS-DFTF was investigated against well known filter based feature selection techniques using various well known benchmark datasets, two popular classification algorithms and four performance evaluation measures. The results of an in-depth experimental analysis noticeably indicate that FS-DFTF based feature selection scheme is better than other filter techniques.

Acknowledgement

The authors would like to thank the Management and Principal of Mepco Schlenk Engineering College (Autonomous), Sivakasi for providing us the state-of-the-art facilities to carry out this proposed research work in the Mepco Research Centre in collaboration with Anna University Chennai, Tamil Nadu, India.

[2] `https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection`

Conflict of Interest Statement

On behalf of all authors, the corresponding author states that there is no conflict of interest.

REFERENCES

[1] Aggarwal, C. C.—Zhai, C.: A Survey of Text Classification Algorithms. In: Aggarwal, C., Zhai, C. (Eds.): Mining Text Data. Springer US, Boston, MA, 2012, pp. 163–222, doi: 10.1007/978-1-4614-3223-4_6.

[2] Anagnostopoulos, I.—Anagnostopoulos, C.—Loumos, V.—Kayafas, E.: Classifying Web Pages Employing a Probabilistic Neural Network. IEE Proceedings – Software, Vol. 151, 2004, No. 3, pp. 139–150, doi: 10.1049/ip-sen:20040121.

[3] Banati, H.—Bajaj, M.: Firefly Based Feature Selection Approach. IJCSI International Journal of Computer Science Issues, Vol. 8, 2011, No. 4, pp. 473–480.

[4] Bracewell, D. B.—Yan, J.—Ren, F.—Kuroiwa, S.: Category Classification and Topic Discovery of Japanese and English News Articles. In: Seda, A., Boubekeur, M., Hurley, T., Mac an Airchinnigh, M., Schellekens, M., Strong, G. (Eds.): Proceedings of the Irish Conference on the Mathematical Foundations of Computer Science and Information Technology (MFCSIT 2006). Electronic Notes in Theoretical Computer Science, Vol. 225, 2009, pp. 51–65, doi: 10.1016/j.entcs.2008.12.066.

[5] Chen, J.—Huang, H.—Tian, S.—Qu, Y.: Feature Selection for Text Classification with Naïve Bayes. Expert Systems with Applications, Vol. 36, 2009, No. 3, Part 1, pp. 5432–5435, doi: 10.1016/j.eswa.2008.06.054.

[6] Chen, R. C.—Hsieh, C. H.: Web Page Classification Based on a Support Vector Machine Using a Weighted Vote Schema. Expert Systems with Applications, Vol. 31, 2006, No. 2, pp. 427–435, doi: 10.1016/j.eswa.2005.09.079.

[7] Cheng, N.—Chandramouli, R.—Subbalakshmi, K. P.: Author Gender Identification from Text. Digital Investigation, Vol. 8, 2011, No. 1, pp. 78–88, doi: 10.1016/j.diin.2011.04.002.

[8] Dada, E. G.—Bassi, J. S.—Chiroma, H.—Abdulhamid, S. M.—Adetunmbi, A. O.—Ajibuwa, O. E.: Machine Learning for Email Spam Filtering: Review, Approaches and Open Research Problems. Heliyon, Vol. 5, 2019, No. 6, Art. No. e01802, doi: 10.1016/j.heliyon.2019.e01802.

[9] Dasgupta, A.—Drineas, P.—Harb, B.—Josifovski, V.—Mahoney, M. W.: Feature Selection Methods for Text Classification. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '07), ACM, 2007, pp. 230–239, doi: 10.1145/1281192.1281220.

[10] Drucker, H.—Wu, D.—Vapnik, V. N.: Support Vector Machines for Spam Categorization. IEEE Transactions on Neural Networks, Vol. 10, 1999, No. 5, pp. 1048–1054, doi: 10.1109/72.788645.

[11] Forman, G.: An Extensive Empirical Study of Feature Selection Metrics for Text Classification. Journal of Machine Learning Research, Vol. 3, 2003, No. 1, pp. 1289–1305.

[12] Günal, S.: Hybrid Feature Selection for Text Classification. Turkish Journal of Electrical Engineering and Computer Sciences, Vol. 20, 2012, No. Sup. 2, pp. 1296–1311.

[13] Gunal, S.—Edizkan, R.: Subspace Based Feature Selection for Pattern Recognition. Information Sciences, Vol. 178, 2008, No. 19, pp. 3716–3726, doi: 10.1016/j.ins.2008.06.001.

[14] Günal, S.—Ergin, S.—Gülmezoğlu, M. B.—Gerek, Ö. N.: On Feature Extraction for Spam E-Mail Detection. In: Gunsel, B., Jain, A. K., Tekalp, A. M., Sankur, B. (Eds.): Multimedia Content Representation, Classification and Security (MRCS 2006). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4105, 2006, pp. 635–642, doi: 10.1007/11848035_84.

[15] Gunal, S.—Gerek, O. N.—Ece, D. G.—Edizkan, R.: The Search for Optimal Feature Set in Power Quality Event Classification. Expert Systems with Applications, Vol. 36, 2009, No. 7, pp. 10266–10273, doi: 10.1016/j.eswa.2009.01.051.

[16] Guru, D. S.—Suhil, M.—Pavithra, S. K.—Priya, G. R.: Ensemble of Feature Selection Methods for Text Classification: An Analytical Study. In: Abraham, A., Muhuri, P. K., Muda, A. K., Gandhi, N. (Eds.): Intelligent Systems Design and Applications (ISDA 2017). Springer, Cham, Advances in Intelligent Systems and Computing, Vol. 736, 2018, pp. 337–349, doi: 10.1007/978-3-319-76348-4_33.

[17] Guyon, I.—Elisseeff, A.: An Introduction to Variable and Feature Selection. Journal of Machine Learning Research, Vol. 3, 2003, pp. 1157–1182.

[18] Hsu, C. W.—Lin, C. J.: A Comparison of Methods for Multiclass Support Vector Machines. IEEE Transactions on Neural Networks, Vol. 13, 2002, No. 2, pp. 415–425, doi: 10.1109/72.991427.

[19] Huang, C. L.—Wang, C. J.: A GA-Based Feature Selection and Parameters Optimization for Support Vector Machines. Expert Systems with Applications, Vol. 31, 2006, No. 2, pp. 231–240, doi: 10.1016/j.eswa.2005.09.024.

[20] Joachims, T.: Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: Nédellec, C., Rouveirol, C. (Eds.): Machine Learning (ECML'98). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1398, 1998, pp. 137–142, doi: 10.1007/bfb0026683.

[21] Kohavi, R.—John, G. H.: Wrappers for Feature Subset Selection. Artificial Intelligence, Vol. 97, 1997, No. 1-2, pp. 273–324, doi: 10.1016/s0004-3702(97)00043-x.

[22] Kumar, M. A.—Gopal, M.: A Comparison Study on Multiple Binary-Class SVM Methods for Unilabel Text Categorization. Pattern Recognition Letters, Vol. 31, 2010, No. 11, pp. 1437–1444, doi: 10.1016/j.patrec.2010.02.015.

[23] Lan, M.—Tan, C. L.—Su, J.—Lu, Y.: Supervised and Traditional Term Weighting Methods for Automatic Text Categorization. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 31, 2009, No. 4, pp. 721–735, doi: 10.1109/tpami.2008.110.

[24] Lee, C.—Lee, G. G.: Information Gain and Divergence-Based Feature Selection for Machine Learning-Based Text Categorization. Information Processing and Management, Vol. 42, 2006, No. 1, pp. 155–165, doi: 10.1016/j.ipm.2004.08.006.

[25] Lewis, D. D.—Ringuette, M.: A Comparison of Two Learning Algorithms for Text Categorization. Third Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, US, 1994, pp. 81–93.

[26] Li, B.—Yan, Q.—Xu, Z.—Wang, G.: Weighted Document Frequency for Feature Selection in Text Classification. 2015 International Conference on Asian Language Processing (IALP), Suzhou, China, 2016, pp. 132–135, doi: 10.1109/IALP.2015.7451549.

[27] Liu, H.—Sun, J.—Liu, L.—Zhang, H.: Feature Selection with Dynamic Mutual Information. Pattern Recognition, Vol. 42, 2009, No. 7, pp. 1330–1339, doi: 10.1016/j.patcog.2008.10.028.

[28] Manning, C. D.—Raghavan, P.—Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, USA, 2008.

[29] Mesleh, A. M.—Kanaan, G.: Support Vector Machine Text Classification System: Using Ant Colony Optimization Based Feature Subset Selection. 2008 International Conference on Computer Engineering and Systems, Cairo, Egypt, 2008, pp. 143–148, doi: 10.1109/icces.2008.4772984.

[30] Rehman, A.—Javed, K.—Babri, H. A.: Feature Selection Based on a Normalized Difference Measure for Text Classification. Information Processing and Management, Vol. 53, 2017, No. 2, pp. 473–489, doi: 10.1016/j.ipm.2016.12.004.

[31] Kim, S.-B.—Han, K.-S.—Rim, H.-C.—Myaeng, S.-H.: Some Effective Techniques for Naive Bayes Text Classification. IEEE Transactions on Knowledge and Data Engineering, Vol. 18, 2006, No. 11, pp. 1457–1466, doi: 10.1109/tkde.2006.180.

[32] Sebastiani, F.: Machine Learning in Automated Text Categorization. ACM Computing Surveys, Vol. 34, 2002, No. 1, pp. 1–47, doi: 10.1145/505282.505283.

[33] Shang, W.—Huang, H.—Zhu, H.—Lin, Y.—Qu, Y.—Wang, Z.: A Novel Feature Selection Algorithm for Text Categorization. Expert Systems with Applications, Vol. 33, 2007, No. 1, pp. 1–5, doi: 10.1016/j.eswa.2006.04.001.

[34] Stamatatos, E.: Author Identification: Using Text Sampling to Handle the Class Imbalance Problem. Information Processing and Management, Vol. 44, 2008, No. 2, pp. 790–799, doi: 10.1016/j.ipm.2007.05.012.

[35] Su, J.—Sayyad-Shirabad, J.—Matwin, S.: Large Scale Text Classification Using Semi-Supervised Multinomial Naive Bayes. Proceedings of the 28th International Conference on Machine Learning (ICML 2011), Bellevue, Washington, USA, 2011, pp. 97–104.

[36] Thirumoorthy, K.—Muneeswaran, K.: Optimal Feature Subset Selection Using Hybrid Binary Jaya Optimization Algorithm for Text Classification. Sādhanā, Vol. 45, 2020, No. 1, Art. No. 201, pp. 1–13, doi: 10.1007/s12046-020-01443-w.

[37] Uysal, A. K.—Gunal, S.: A Novel Probabilistic Feature Selection Method for Text Classification. Knowledge-Based Systems, Vol. 36, 2012, pp. 226–235, doi: 10.1016/j.knosys.2012.06.005.

[38] Xu, Y.—Wang, B.—Li, J.—Jing, H.: An Extended Document Frequency Metric for Feature Selection in Text Categorization. In: Li, H., Liu, T., Ma, W. Y., Sakai, T., Wong, K. F., Zhou, G. (Eds.): Information Retrieval Technology (AIRS

2008). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4993, 2008, pp. 71–82, doi: 10.1007/978-3-540-68636-1_8.

[39] Yang, J.—Honavar, V.: Feature Subset Selection Using a Genetic Algorithm. IEEE Intelligent Systems and Their Applications, Vol. 13, 1998, No. 2, pp. 44–49, doi: 10.1109/5254.671091.

[40] Wu, Y.—Zhang, A.: Feature Selection for Classifying High-Dimensional Numerical Data. Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), 2004, Vol. 2, pp. II–II, doi: 10.1109/cvpr.2004.1315171.

[41] Yu, B.—Zhu, D.-H.: Combining Neural Networks and Semantic Feature Space for Email Classification. Knowledge-Based Systems, Vol. 22, 2009, No. 5, pp. 376–381, doi: 10.1016/j.knosys.2009.02.009.

Thirumoorthy Karpagalingam received his M.E. degree from the Arulmigu Kalasalingam College of Engineering, Krishnankoil, Tamilnadu, India and his B.E. degree from the Kamaraj College of Engineering and Technology, India. He worked as Assistant Professor in Computer Science and Engineering Department of the Mepco Schlenk Engineering College, Sivakasi for 12 years and currently he is pursuing Ph.D. in Mepco Schlenk Engineering College, Sivakasi, India under Anna University, Chennai, India. His research areas include text mining. His publications have appeared in various leading journals and conferences.

Muneeswaran Karuppaiah is presently Senior Professor in the Department of Computer Science and Engineering at Mepco Schlenk Engineering College, Sivakasi, India. He completed his doctorate in M.S. University, Tirunelveli, India, his post graduate degree from PSG College of Technology, Coimbatore, India and Bachelor degree from Thiagarajar College of Engineering, Madurai, India. He has nearly 33 years of teaching experience. He has published nearly 52 papers in reputed international journals with 531 citations and 95 papers in international and national conferences. He has published a book "Compiler Design" in Oxford University Press. He is also a life member of organizations such as the Computer Society of India (CSI), Indian Society for Technical Education (ISTE) and Institution of Electronics and Telecommunication Engineers (IETE).

Computing and Informatics, Vol. 39, 2020, 907–924, doi: 10.31577/cai_2020_5_907

HIERARCHICAL TEXT CLASSIFICATION USING CNNS WITH LOCAL APPROACHES

Milan Krendzelak, Frantisek Jakab

Technical University of Košice
Faculty of Electrical Engineering and Informatics
Department of Computers and Informatics
Letná 9, 040 01 Košice, Slovakia
e-mail: krendzelak.m@gmail.com, frantisek.jakab@tuke.sk

Abstract. In this paper, we discuss the application of convolutional neural networks (CNNs) for hierarchical text classification using local top-down approaches. We present experimental results implementing a local classification per node approach, a local classification per parent node approach, and a local classification per level approach. A 20Newsgroup hierarchical training dataset with more than 20 categories and three hierarchical levels was used to train the models. The experiments involved several variations of hyperparameters settings such as batch size, embedding size, and number of available examples from the training dataset, including two variation of CNN model text embedding such as static (stat) and random (rand). The results demonstrated that our proposed use of CNNs outperformed flat CNN baseline model and both the flat and hierarchical support vector machine (SVM) and logistic regression (LR) baseline models. In particular, hierarchical text classification with CNN-stat models using local per parent node and local per level approaches achieved compelling results and outperformed the former and latter state-of-the-art models. However, using CNN with local per node approach for hierarchical text classification underperformed and achieved worse results. Furthermore, we performed a detailed comparison between the proposed hierarchical local approaches with CNNs. The results indicated that the hierarchical local classification per level approach using the CNN model with static text embedding achieved the best results, surpassing the flat SVM and LR baseline models by 7 % and 13 %, surpassing the flat CNN baseline by 5 %, and surpassing the h-SVM and h-LR models by 5 % and 10 %, respectively.

Keywords: Hierarchical text classification, convolutional neural network, local top-down approach

Mathematics Subject Classification 2010: 68-U01

1 INTRODUCTION

Various methods exist for solving the hierarchical text classification (HTC) task, which is primarily based on how hierarchical relationships are utilized. Flat classification treats a flattened taxonomy as a set of unique classes that represent its hierarchy. Therefore, a binary classifier is usually trained for each class to discriminate it from the remaining classes. Although the flat classification approach is well known for its efficiency and simplicity when handling small-sized and well-balanced datasets, its performance suffers when the dimensions of the classes to be predicted not only increase but also become hierarchically interdependent. Due to the invocation of binary classifiers for all trained nodes, computation becomes time-consuming and costly [1].

The application of hierarchically organized categories into a taxonomy has become the most frequent method of organizing large quantities of data. For instance, enterprise customer care, product knowledge bases, online self-help centers, and e-learning systems. The most frequent strategy is to flatten a taxonomy so that all training datasets belong only to leaf classes. However, organizing the training datasets in this way does not mimic real-world examples very effectively. The connection between classes is lost because of the flattening process; thus, it is not possible to forecast the parent category of a new case [2].

However, in the context of HTC hierarchical relationships between parent and children elements, derived from a taxonomy of classes, should be considered either for training or predicting phases or both. In this case, the difference between different approaches to tackle the HTC task is the way the training dataset is leveraged. Currently, there are well-known hierarchical approaches known as local and global approaches. A hierarchical local approach is further divided into a local per node approach, local per parent node approach, and a local per level approach. Then the local approach involves splitting the hierarchical structure into several smaller structures for training local learning models; the global approach consumes the entire hierarchy of classes at once for training the global learning model [3].

2 RELATED WORK

There is a number of studies that focus on solving hierarchical text classification. For example, a well-known hierarchical top-down approach with level-based support vector machine models for text classification has been suggested by Sun and Lim [4]. Similarly, Sokolov et al. proposed a model for ontology term forecasting by explicitly simulating a construction hierarchy using kernel techniques for structured output [8]. Cerri et al. proposed an approach for hierarchical multi-label text classification that involves training a multi-layer perceptron for each level of the classification hierarchy [9]. The predictions generated by a neural network in a given level serve as inputs to the neural network responsible for the forecast in the following level. Their method was evaluated against several datasets with promising results.

Within the context of neural networks, Kurata et al. proposed a strategy for initializing neural networks' hidden output by considering multi-label co-occurrence. Their method treats a number of neurons in the final hidden layer as committed neurons for every pattern of tag co-occurrence [10]. In addition, there have been several important studies that proposed the inclusion of multi-label co-occurrence into loss functions, such as pairwise standing loss by Zhang and Zhou [11]. Furthermore, in more recent work, Nam et al. [12] reported that binary cross-entropy can outperform pairwise ranking reduction by minding rectified linear units (ReLUs) for nonlinearity.

3 LOCAL CLASSIFICATION APPROACHES

The hierarchical local classification approach deals with the local cross-section of hierarchically organized classes in order to consider information about parent-child and sibling relationships during the training phase. Based on different methods of applying the cross-section to local information extraction, the local classification approach is further divided into three subcategories.

3.1 Local Classifier per Node

The local classifier per node (LCN) approach dictates that a binary classifier ψ_n is learned for each node $n \in N$ except for the root node R in the hierarchy H, as illustrated in Figure 1. The dashed squares represent binary classifiers that are assembled into top–down manner execution.

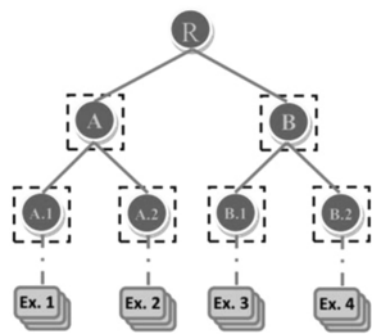


Figure 1. Local classifier per node

The training of a binary classifier at a node is performed by feeding the model with positive and negative examples. With respect to the LCN, all examples belonging to the n^{th} node and its descendants are considered positive training examples, while examples belonging to the n^{th} node siblings and their descendants are consid-

ered negative examples. The binary classifier is formulated as follows:

$$\sum_{i=1}^{N} \mathscr{L}(w_l, x(i), y(i)) + \lambda ||w_l||_2^2. \tag{1}$$

This classifier attempts to minimize the weight vectors for each label l, where $\lambda > 0$ is the penalty parameter, $\mathscr{L}$ denotes the loss function (e.g., hinge loss or logistic loss), and $||_2^2$ denotes the squared ℓ_2-norm. To predict an unknown test instance, the algorithm generally proceeds in a top-down manner, starting at the root and recursively selecting the best children until it reaches a terminal node that belongs to the set of leaf categories $\mathscr{L}$, which is the final predicted node.

The strategy described above is the one most commonly found in the literature. However, there exist additional strategies with different fine-tuning methods for annotating training data to differentiate among positive and negative examples.

3.2 Local Classifier per Parent Node

The local classifier per parent node (LCPN) approach states that a multi-class classifier is learned for each parent node $p \in N$ in the hierarchy $\mathscr{H}$, as illustrated in Figure 2. The dashed squares in the figure represent multi-class classifiers.

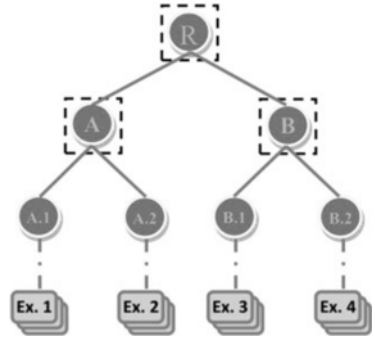


Figure 2. Local classifier per node

Like the LCN, the goal of the LCPN is to learn classifiers that can effectively discriminate between siblings. To train the classifier at each parent node N, we use the examples from its descendants, in which each of the children categories $C(N)$ of parent node N corresponds to different classes. The multi-class classifier is formulated as follows:

$$\text{minimize} \frac{1}{N} \sum_{i=1}^{N} \xi_i + \lambda \sum_{l=1}^{L} ||w_l||_2^2, \tag{2}$$

assuming that

$$w_{l_i}^T x(i) - w_l^T x(i) \geq 1 - \xi_i, \quad \forall l \in L - l_i, \forall i \in [1, 2, ..., N]$$

and

$$w_{l_i}^T x(i) - w_l^T x(i) >= 1 - \xi_i.$$

This classifier attempts to minimize the weight vectors, where $\lambda > 0$ is the penalty parameter, $\mathscr{L}$ denotes the loss function (e.g., hinge loss or logistic loss), ξ_i denotes the slack variables, and $||_2^2$ denotes the squared ℓ_2-norm.

3.3 Local Classifier per Level

In the local classifier per level (LCL) approach, a multi-class classifier is learned for every level in the hierarchy, as illustrated in Figure 3. To train the classifier at each level, examples from the nodes are used individually for each level along with its descendants. It should be noted that nodes at the same level do not overlap and correspond to distinct classes. Prediction is performed by selecting the best node at each level in the hierarchy.

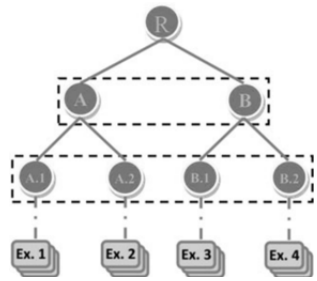


Figure 3. Local classifier per level

Because classifiers at each level make independent predictions, it is possible that this approach may result in vertical inconsistency in prediction. For this strategy to be useful, a post-processing measure is employed to solve inconsistent predictions, if necessary.

4 HIERARCHICAL PERFORMANCE EVALUATION

4.1 Flat Evaluation Metrics

As metrics, we use the standard micro-F_1 (μF_1) score and macro-F_1 (MF_1) score to evaluate the performance of various methods. To compute μF_1, we sum the category-specific true positives (TP_c), false positives (FP_c), and false negatives

(FN_c). Then, the definition shall be for different categories define micro-F_1 as follows:

$$\mu F_1 = \frac{2\ P\ R}{P + R} \tag{3}$$

where P is precision and R is recall, defined as follows:

$$P = \frac{\sum_{c \in \mathscr{L}} TP_c}{\sum_{c \in \mathscr{L}(TP_c + FP_c)}}, \tag{4}$$

$$R = \frac{\sum_{c \in \mathscr{L}} TP_c}{\sum_{c \in \mathscr{L}(TP_c + FN_c)}}. \tag{5}$$

The MF_1 score, which gives equal weight to all categories so that the average score is not skewed in favor of the larger categories, is defined as follows:

$$MF_1 = \frac{1}{|\mathscr{L}|} \sum_{cin\mathscr{L}} \frac{2P_cR_c}{P_c + R_c} \tag{6}$$

where $|\mathscr{L}|$ is the number of leaf categories, and P_c and R_c are defined as follows:

$$P_c = \frac{TP_c}{TP_c + FP_c}, \tag{7}$$

$$R_c = \frac{TP_c}{TP_c + FN_c}. \tag{8}$$

4.2 Hierarchical Evaluation Metrics

With respect to HTC performance, hierarchical metrics should consider the hierarchical distance between the true class and predicted class. The principle is to penalize misclassification differently from flat metrics, which penalize each misclassified example equally. Generally, misclassifications that are closer to the actual class are penalized less than misclassifications which are further from it with respect to the hierarchy.

The hierarchical metrics include the hierarchical F_1 (hF_1) score, hierarchical precision P (hP), hierarchical recall R (hR), and tree-induced error (TE), which are defined as follows:

$$hF_1 = \frac{2\ hP\ hR}{hP + hR}, \tag{9}$$

$$TE = \frac{1}{N} \sum_{i=1}^{N} \delta(\hat{y}_i, y_i) \tag{10}$$

where hP and hR are defined as follows:

$$hP = \frac{\sum_{i=1}^{N} |A(\hat{y}_i) \cap A(y_i)|}{\sum_{i=1}^{N} |A(\hat{y}_i)|}, \tag{11}$$

$$hR = \frac{\sum_{i=1}^{N} |A(\hat{y}_i) \cap A(y_i)|}{\sum_{i=1}^{N} |A(y_i)|}. \tag{12}$$

Here, $A(\hat{y}_i)$ and $A(y_i)$ are the set of ancestors of the predicted and true labels, respectively, including the class itself but not the root node. $\delta(\hat{y}_i, y_i)$ represents the length of the undirected path between categories $\hat{y}_i$ and y_i in the tree.

5 TOP-DOWN HIERARCHICAL ENSEMBLE

A top-down ensemble of prediction evaluation is one of the most efficient approaches for solving the HTC task with binary classifiers, either implemented as CNN, SVN or LR models [8]. The principle of this approach is to recursively evaluate a prediction from the top of the hierarchy down to the leaf, traversing each node on the path, as illustrated in Algorithm 1. At each step, the node with the highest prediction score is selected. This process repeats recursively until the last leaf node is reached, which corresponds to a certain class and is usually located at the bottom.

Result: Repeat recursively until last leaf node is reached
initialization n := Root;
while $n \notin L$ **do**
  $n := \arg\max_{q \in C(n)} f_q(x)$;
end
return n;

Algorithm 1: Local top-down approach

Top-down methods are popular for large-scale problems due to their computational advantages when only a subset of classes in the appropriate path is considered in the prediction phase. In addition, these methods minimize problems related to prediction inconsistencies because the best child node is selected at each level of the path. Top-down methods have been successfully utilized to resolve HTC problems either for learning or training and prediction phases.

A major disadvantage of top-down technique that results in poor classification performance is error propagation, namely, the compounding of errors from misclassifications at higher levels that cannot be corrected at the next lower levels. This problem can be relieved to a certain extent by shifting the hierarchy to temper the level of deformation. It is important to note that this top-down technique is applied only for binary classifiers.

6 CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNNs) have been adopted from the field of computer vision, in which they have been shown to provide state-of-the-art results with default baseline hyperparameter settings.

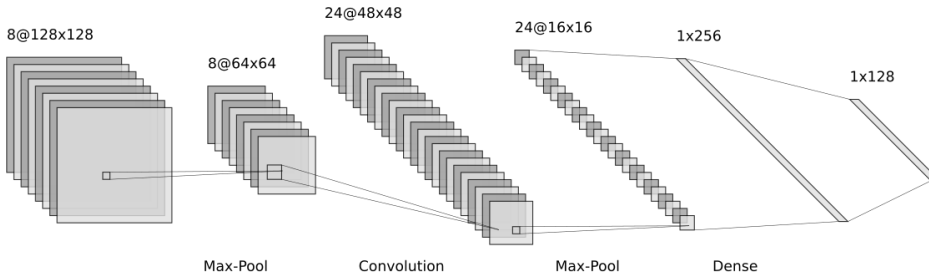


Figure 4. Convolution neural network (CNN)

Let $x_i \in R^k$ be the k-dimensional word vector relevant to the i^{th} word in a sentence. A sentence of length n (padded if required) is represented as follows:

$$x_{1:n} = x_1 \oplus x_2 \oplus \dots x_n \tag{13}$$

where $\oplus$ is the concatenation operator. Furthermore, let $x_{i:i+j}$ refer to the concatenation of words $x_i, x_{i+1}, \dots x_{i+j}$. Applying a filter $w \in R^h k$ in strides defines a convolution operation, which is applied to a selected window of h terms to compute a new feature. Therefore, feature c_i is produced for each window of words $x_{i:i+h-1}$ as follows:

$$c_i = f(w\ x_{i:i+h-1} + b). \tag{14}$$

Let $b \in R$ be a bias and f a non-linear function, for example, a hyperbolic tangent or relu function. Then, a filter is used to stride through each possible window in the given sentence $\{x_{1:h}, x_{2:h+1}, \dots x_{n-h+1:n}\}$ to create a feature map, such as

$$c = [c_1, c_2, c_3, \dots c_{n-h+1}] \tag{15}$$

where $c \in R^{n-h+1}$. Thereafter, a max-overtime pooling operation is applied over the set of feature maps to select a maximum value $C = \max\{c\}$ as the feature. The general principle is to be able to capture the most important set of features for all feature maps.

The above process describes how one feature is extracted from one filter. However, the model uses multiple filters, typically with varying window sizes, to extract multiple features. Therefore, these features form next to the last layer, which is directly followed by a fully connected softmax layer whose output is the probability distribution over the labels.

6.1 Convolution

A 1D convolution is an operation between a vector of weights m, where $m \in R^m$, and a vector of input sequences s, where $s \in R^s$. Vector m is the filter of the convolution. Let s be an input sentence and $s_i \in R$ be a single feature value associated with the i^{th} word in a sentence. Then, a 1D convolution produces the dot product of vector m with each n-gram in the sentence s to obtain another sequence c as follows:

$$c_j = m^T S_{j-m+1:j}. \tag{16}$$

Equation (16) describes two possible types of convolution – narrow and wide – depending on the range of index j. The narrow type of convolution requires that $s \geq m$, and yields a sequence $c \in R^{s-m+1}$ with j ranging from m to s. The wide type of convolution does not constrain m or s, and yields sequence $c \in R^{s+m-1}$, where index j ranges from 1 to $s+m-1$. Values s_i outside of the range are considered to be zero, where $i < 1$ or $i > s$.

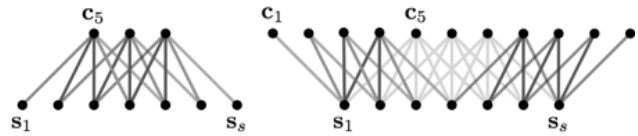


Figure 5. Narrow and wide convolutional layers

The result of a narrow convolution is a subsequence of the results of the wide convolution. The two types of one-dimensional convolutions are illustrated in Figure 5. The trained weights in filter m correspond to a linguistic feature that learns to recognize a particular class of n-grams. These n-grams have size $n \leq m$, where m is the width of the filter. Applying the weights m in a wide convolution has several advantages over applying them in a narrow convolution. A wide convolution ensures that all weights in the filter reach the entire sentence, including words at margins and paddings.

6.2 Feature Selection

One of the advantages of a CNN over other neural networks is that it is designed to automatically extract features from the given text corpora. It does so by applying convolutional layers in a specific, predefined manner as described in Section 6.1. These extracted feature maps are more efficient and less error-prone than manually constructed ones. In addition, they ensure a high level of accuracy in capturing the most important details for given examples of text data.

In general, convolution layers can be considered a feature extractor whose output is fed into a fully connected layer for the purpose of simple decision-making, such as classification or ranking. Because a CNN creates local features for each word in

a sentence, it is possible to combine or stack features to produce a global feature vector. Several aspects of a CNN's ability to create and extract text features are as follows:

- A CNN internally creates features that can be extracted and used as input for other custom-defined internal layers or external models.
- A CNN automatically creates features that do not rely on a hand-crafted process. It adapts well to the specifics of a training dataset in a supervised manner.
- A hierarchy of local features is considered during feature creation; therefore, the CNN captures the context effectively.

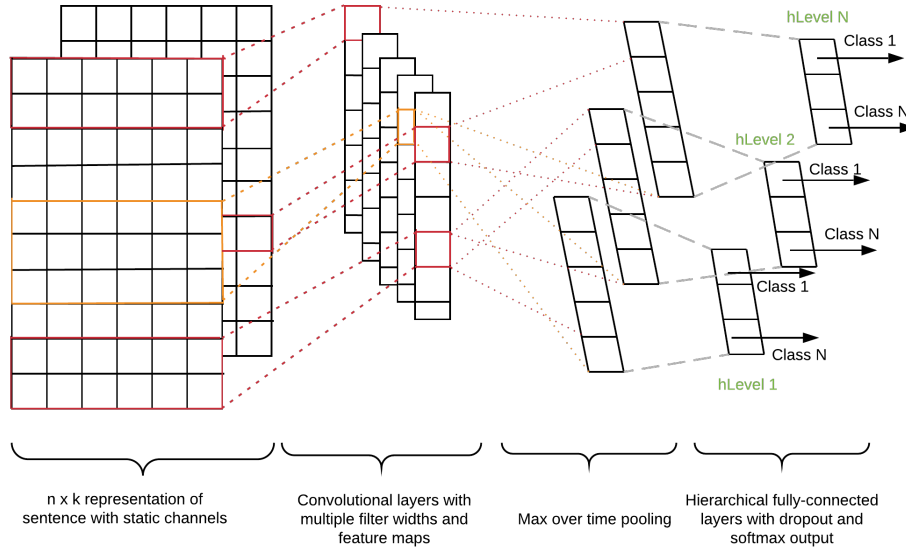


Figure 6. Conceptual diagram of experimental CNN for local per level classification

7 EXPERIMENTAL SETUP

To provide a comprehensive assessment of the use of CNNs for hierarchical text classification using local approaches, we conducted three independent experiments and observed, measured, and compared the outcomes.

It should be noted that the following constrains were taken into consideration and applied throughout the experimentation:

- The training dataset outlined in Figure 7 was used; thus, the results could be compared with each other and with previously reported applications of h-SVM and h-LR models using identical local approaches operated in a top-down fashion.

- The same variation of the CNN baseline model was used; thus, the outcome did not depend on the model specifics, but rather on the effectiveness of the local approach as a strategy.
- For all experiments in this study, we used 300-dimensional word vectors trained by Mikolov et al. on roughly 100 billion words from Google News.

7.1 Training Dataset

The 20Newsgroup dataset contained an average of 20 000 e-news items from more than 20 different categories hierarchically ordered with three hierarchical levels. The total training dataset contained 11 314 examples.

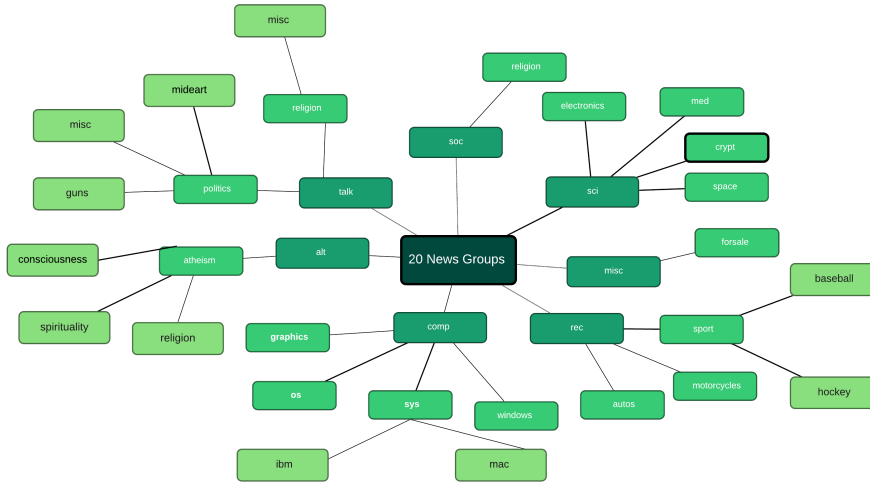


Figure 7. 20Newsgroup training dataset hierarchy

The category size corresponded to the number of available examples per category, as outlined in Table 1. The training data was preprocessed, aggregated for each hierarchy level, and counted for the number of available examples per category.

7.2 CNN Variants

One difference among the many recent studies on word-based CNNs for text classification is the choice of using pretrained or end-to-end learned word representations. In our experiments, we used two variants of the CNN baseline model that differed in the initialization of embedding.

- CNN-rand which learns the text embedding from scratch during the training.
- CNN-stat which is initialized with pretrained word2vec.

7.3 Hyperparameters and training

- Optimizer. For all experiments, we trained our model's parameters with the Adam optimizer initialized with a learning rate of 0.001.
- Batch sizes. We experimented with different batch sizes, such as 64, 128, and 192, as we wished to observe the impact of different values on model performance.
- Embedding sizes. The default embedding size was 128. However, to obtain an improved understanding of how this hyperparameter value affects performance, we experimented with additional settings, such as 256 and 300.
- Number of filters and their sizes. The presented CNN model contained three convolutional layers, each with three different filters. Each filter had a size of 3, 4, and 5.

8 EXPERIMENTS WITH LOCAL APPROACHES

8.1 LCN Experiment

The hierarchical LCN approach consists of binary classifiers, and each classifier is built independently for each node. In this case, all categories listed in Table 1 are considered nodes. For each node, we trained one CNN classifier. In total, we had 27 CNN classifiers corresponding to 27 categories.

Category	Size	Category	Size
comp	2 936	alt-atheism	480
comp.graphics	584	soc-religion	599
comp.os	591	sci	2 373
comp.windows	593	sci.crypt	595
comp.sys	1 168	sci.electronics	591
comp.sys.ibm	590	sci.med	594
comp.sys.mac	578	sci.space	593
rec	2 389	talk	1 952
rec.autos	594	talk.religion-misc	377
rec.motorcycles	598	talk.politics	1 575
rec.sport	1 197	talk.politics.guns	546
rec.sport.baseball	597	talk.politics.mideast	564
rec.sport.hockey	600	talk.politics.misc	465
misc-forsale	585	TOTAL	11 314

Table 1. 20Newsgroup text analysis of training dataset

To perform prediction, top-down sequential evaluation was performed, and at each node, a binary decision was made regarding which classifier to execute next in the evaluation chain. The training dataset for each node was carefully manually crafted in such a way that all examples belonging to the n^{th} node and its descendants

were considered positive training examples and examples belonging to the siblings of the n^{th} node and their descendants were considered negative examples.

8.2 LCPN Experiment

The hierarchical LCPN was implemented as a multi-class classifier for each parent node in the given taxonomy. For our experiment, the nodes that were considered are listed in Table 2.

Parent Node	Size	Classes
ROOT	11 314	comp, rec, sci, talk, misc-forsale, alt-atheism, soc-religion
comp	2 936	comp.graphics, comp.os, comp.windows, comp.sys
comp.sys	1 168	comp.sys.ibm, comp.sys.mac
rec	2 389	rec.autos, rec.motocycles, rec.sport
rec.sport	1 197	rec.sport.baseball, rec.sport.hockey
sci	2 373	sci.crypt, sci.electronics, sci.med, sci.space
talk	1 952	talk.religion-misc, talk.politics
talk.politics	1 575	talk.politics.guns, talk.politics.mideast, talk.politics.misc

Table 2. Training data for LCPN

We trained eight independent multi-class CNN classifiers for each parent node, including the ROOT node. Each classifier learned from the subset of training data created in such a way that only examples belonging to the parent nodes and their descendants child nodes were selected as positive examples. We did not feed negative examples into the model because neural networks are usually trained with positive examples.

8.3 LCL Experiment

The hierarchical LCL was implemented as a multi-class classifier for every level in the given taxonomy. We thus had three levels, as listed in Table 3. Each level contained only nodes belonging to a certain hierarchical level.

We trained each of the three multi-class CNN classifiers for every level. A prediction evaluation was performed in a top-down fashion starting from the first level and iterating through the remainder of the levels. Training of the level-based classifier was performed by feeding the model with examples from the descendant nodes belonging to their level-based parents.

The conceptual diagram of CNN model with LCL approach is listed in Figure 6. It can be observed that this model implements only 3 multi-class classifiers, each classifier represents exactly one of the hierarchical levels as listed in Table 3 in such way that there is no overlap among categories and different levels.

The final dense layer in CNN baseline contains a single node for each target class in the model. However, in order to represent LCL approach using CNN, the baseline model dense layer is modified and implemented 3 fully-connected layers.

Level	Size	Category
1	11 314	alt-atheism, comp, rec, misc-forsale, soc-religion-christian, sci, talk
2	9 650	comp.graphics, comp.os, comp.sys, comp.windows, rec.autos, rec.motorcycles, rec.sport, sci.crypt, sci.electronics, sci.med, sci.space, talk.religion-misc, talk.politics
3	3 940	comp.sys.ibm, comp.sys.mac, rec.sport.baseball, rec.sport.hockey, talk.politics.guns, talk.politics.mideast, talk.politics.misc

Table 3. LCL training datset

9 EVALUATION OF EXPERIMENTS

To determine the effects of using CNN models in hierarchical text classification, we performed multiple tests to experiment with a different set of hyperparameters. We conducted three major experiments with the proposed hierarchical local approaches and compared our results with available flat LR and SVM baseline models and state-of-the-art models, such as h-LR and h-SVM. One of our goals was to demonstrate that CNNs can be successfully used for solving hierarchical text classification problems in a more effective and simplified manner than existing methods.

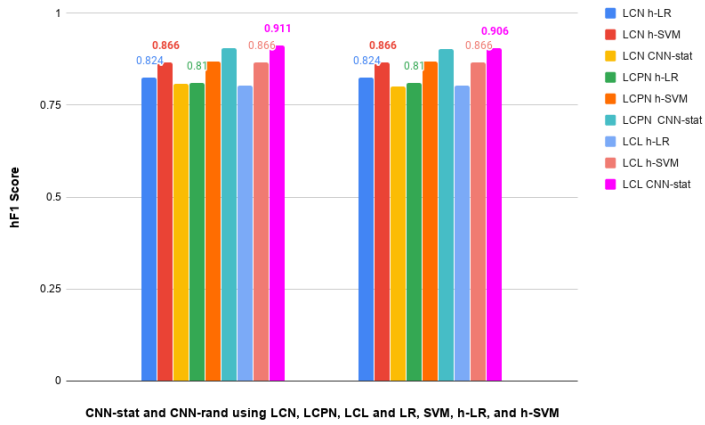


Figure 8. Benchmark of experimental CNN-rand and CNN-stat using LCN, LCPN and LCL with h-LR and h-SVM using LCN, LCPN and LCL

Of the proposed methods, we determined that the LCN hierarchical approach was the most complex. We observed that different variations of examples with different dataset sizes had a direct impact on the accuracy of the binary classifier. It thus requires additional effort during the pre-processing phase to prepare positive

and negative training examples arbitrarily selected from the dataset. Moreover, top-down prediction requires the evaluation of the final prediction to be proceeded in a node-chained manner, emulating the path of a hierarchical taxonomy. There is no mechanism available in the baseline model to mitigate the error propagated from the previous node prediction.

Our empirical observations of the LCPN approach indicated that it was less complex than the previous LCN approach mentioned above. The LCPN requires the construction of only eight multi-class classifiers whose predictions are chained in a top-down fashion to perform hierarchical prediction. Training a multi-class neural network is quite different than training a binary classifier. We observed that for this approach, less effort was required during the pre-processing of the training examples required to train a model with hierarchically categorical classes. This was due to the fact that a CNN is capable of consuming raw training data and does not require positive and negative samples.

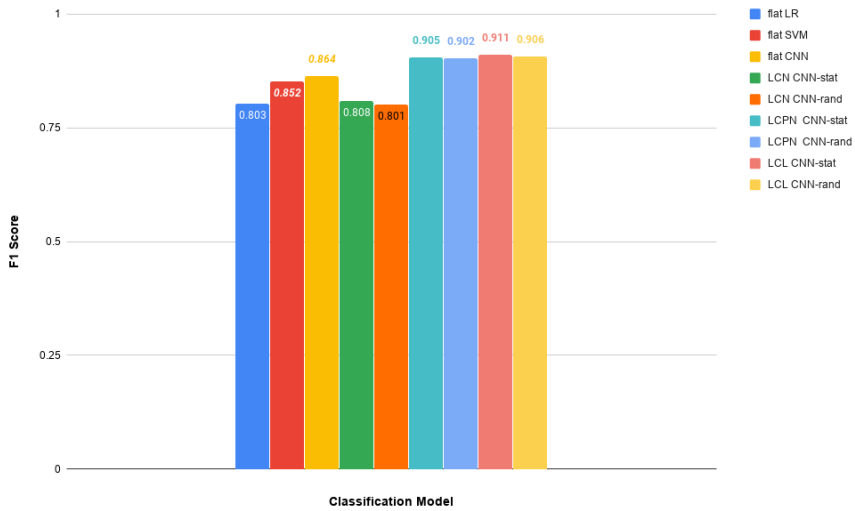


Figure 9. Benchmark of experimental CNN-stat and CNN-rand using LCN, LCPN and LCL with flat LR, SVM, and CNN baseline models

We found that the best results were achieved by using the LCL approach implemented by CNN-stat model with a hF_1 score of 0.911 trained with a dataset of size 10 500. For the CNN-rand model using LCL approach, the best results were achieved with a hF_1 score of 0.906. However, we observed that the latter model, CNN-rand, required more training data to tune its performance and was unable to outperform CNN-stat.

10 CONCLUSION

In this study, we proposed a novel application of a CNN for solving the hierarchical text classification problem using hierarchical local classification approaches. We demonstrated that hierarchical local approaches with CNN models achieved results superior to those of the flat LR and SVM baseline, which results were reported by Song et al. [14]. Moreover, additionally, experimental results achieved by proposed use of CNNs surpassed flat CNN baseline model by 5 %, which results were reported by Prakhya et al. [15].

The results confirmed that the CNN-stat LCL approach achieved the best results among the tested local approaches, furthermore, outperforming flat SVM baseline model by 7 % and flat LR baseline model by 13 %. In regards to h-SVM and h-LR models, these were outperformed by CNN-stat LCL approach by 5 % and h-LR by 10 %, as can be observed in Table 9. Moreover, the CNN-stat LCL approach required only 3 multi-class CNN classifiers, compared to 27 binary classifiers required by LCN approach and 8 multi-class classifiers required by LCL approach.

Additionally, we observed that the CNN-stat model, which has the text embedding layer initialized with pre-trained word2vec, outperformed the CNN-rand model most of the time, except only observed once. This is due primarily to the fact that pre-trained text embedding has a more precise and comprehensive representation of words than a randomly initialized embedding layer learned during the training phase.

REFERENCES

[1] Zimek, A.—Buchwald, F.—Frank, E.—Kramer, S.: A Study of Hierarchical and Flat Classification of Proteins. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Vol. 7, 2010, No. 3, pp. 563–571, doi: 10.1109/tcbb.2008.104.

[2] Babbar, R.—Partalas, I.—Gaussier, E.—Amini, M. R.: On Flat Versus Hierarchical Classification in Large-Scale Taxonomies. In: Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 26 (NIPS 2013), Vol. 2, 2013, pp. 1824–1832.

[3] Krendzelak, M.—Jakab, F.: Approach for Hierarchical Global All-In Classification with Application of Convolutional Neural Networks. 2018 16$^{\text{th}}$ International Conference on Emerging eLearning Technologies and Applications (ICETA 2018), 2018, pp. 317–322, doi: 10.1109/iceta.2018.8572074.

[4] Sun, A.—Lim, E.—Ng, W.: Performance Measurement Framework for Hierarchical Text Classification. Journal of the American Society for Information Science and Technology, Vol. 54, 2003, No. 11, pp. 1014–1028, doi: 10.1002/asi.10298.

[5] Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research, Vol. 7, 2006, No. 1, pp. 1–30.

[6] Dumais, S.—Chen, H.: Hierarchical Classification of Web Content. Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '00), 2000, pp. 256–263, doi: 10.1145/345508.345593.

[7] Wang, K.—Zhou, S.—He, Y.: Hierarchical Classification of Real-Life Documents. Proceedings of the 2001 SIAM International Conference on Data Mining, 2001, doi: 10.1137/1.9781611972719.22.

[8] Sokolov, A.—Funk, C.—Graim, K.—Verspoor, K.—Ben-Hur, A.: Combining Heterogeneous Data Sources for Accurate Functional Annotation of Proteins. BMC Bioinformatics, Vol. 14, 2013, No. 3, Art. No. S10, doi: 10.1186/1471-2105-14-s3-s10.

[9] Cerri, R.—Barros, R. C.—de Carvalho, A. C. P. L. F.: Hierarchical Multi-Label Classification Using Local Neural Networks. Journal of Computer and System Sciences, Vol. 80, 2014, No. 1, pp. 39–56, doi: 10.1016/j.jcss.2013.03.007.

[10] Kurata, G.—Xiang, B.—Zhou, B.: Improved Neural Network-Based Multi-Label Classification with Better Initialization Leveraging Label Cooccurrence. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2016, pp. 521–526, doi: 10.18653/v1/n16-1063.

[11] Zhang, M. L.—Zhou, Z. H.: Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization. IEEE Transactions on Knowledge and Data Engineering, Vol. 18, 2006, No. 10, pp. 1338–1351, doi: 10.1109/tkde.2006.162.

[12] Nam, J.—Kim, J.—Mencia, E. L.—Gurevych, I.—Fürnkranz, J.: Large-Scale Multi-Label Text Classification – Revisiting Neural Networks. In: Calders, T., Esposito, F., Hüllermeier, E., Meo, R. (Eds.): Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2014). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 8725, 2014, pp. 437–452, doi: 10.1007/978-3-662-44851-9_28.

[13] Kim, Y.: Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751, doi: 10.3115/v1/d14-1181.

[14] Song, Y.—Roth, D.: On Dataless Hierarchical Text Classification. AAAI Publications, Twenty-Eighth AAAI Conference on Artificial Intelligence, Québec, Canada, 2014, pp. 1579–1585.

[15] Prakhya, S.—Venkataram, V.—Kalita, J.: Open Set Text Classification Using Convolutional Neural Networks. International Conference on Natural Language Processing, USA, 2017.

Milan Krendzelak is Tech Lead of Web Solutions Engineers team at Google Inc. that delivers innovative solutions to help drive revenue and make Google's global sales field more efficient. As Web Solutions Engineer, his responsibilities include prototyping proofs of concept, developing and supporting tools, and enhancing core products to meet the needs of his sales field. He earned a master of computer science degree in 2004 at the Department of Computers and Informatics at the Technical University of Košice, Slovakia. Currently, he is continuing his Ph.D. research in Hierarchical Text Classification at the Department of Computers and Informatics at the Technical University of Košice, Slovakia. He has published more than six scientific publications.

Frantisek Jakab is Director of the University Science Park TECHNICOM and Head of the Computer Networks Laboratory (`www.cnl.sk`) that he created during his career at the Department of Computers and Informatics at the Technical University of Košice, Slovakia. He graduated from the Faculty of Computer Science and Electrical Engineering at the St. Petersburg Institute of the Electrical Engineering in the field of System Engineering (Russian Federation). Main areas of his research activities: computer networks, which is a new form of multimedia-based communication (video conferences, IP streaming). He is renowned author of more than 200 scientific publications and textbooks.

Computing and Informatics, Vol. 39, 2020, 925–951, doi: 10.31577/cai_2020_5_925

ASYNCHRONOUS SPIKING NEURAL P SYSTEMS WITH MULTIPLE CHANNELS AND SYMBOLS

Wenmei Yi, Zeqiong Lv, Hong Peng*

School of Computer and Software Engineering
Xihua University, Chengdu, 610039, China
e-mail: wenmeiyee@foxmail.com, ph.xhu@hotmail.com

Xiaoxiao Song, Jun Wang

School of Electrical Engineering and Electronic Information
Xihua University, Chengdu, 610039, China
e-mail: wj.xhu@hotmail.com

Abstract. Spiking neural P systems (SNP systems, in short) are a class of distributed parallel computation systems, inspired from the way that the neurons process and communicate information by means of spikes. A new variant of SNP systems, which works in asynchronous mode, asynchronous spiking neural P systems with multiple channels and symbols (ASNP-MCS systems, in short), is investigated in this paper. There are two interesting features in ASNP-MCS systems: multiple channels and multiple symbols. That is, every neuron has more than one synaptic channels to connect its subsequent neurons, and every neuron can deal with more than one type of spikes. The variant works in asynchronous mode: in every step, each neuron can be free to fire or not when its rules can be applied. The computational completeness of ASNP-MCS systems is investigated. It is proved that ASNP-MCS systems as number generating and accepting devices are Turing universal. Moreover, we obtain a small universal function computing device that is an ASNP-MCS system with 67 neurons. Specially, a new idea that can solve "block" problems is proposed in INPUT modules.

Keywords: Membrane computing, spiking neural P systems, asynchronous systems, multiple channels, multiple symbols, Turing universality

1 INTRODUCTION

Abstracted from the structure and functioning of living cells as well as the cooperation of cells in tissue, organs and biological nervous systems, membrane computing is a class of distributed and parallel computation systems [1, 2], known as P systems. There are three main types of P systems: cell-like P systems, tissue-like P systems and neural-like P systems. Abstracted from distinct biological cell mechanisms, a lot of P systems and variants have been proposed, for example, tissue-like P systems [3], population P systems [4], P colonies [5], spiking neural P systems [6], and the latest works can be found on the membrane computing website (`http://ppage.psystems.eu`). In terms of computational theory, most of P systems have been proven to be Turing universal, and some NP-hard problems have been solved in a feasible time [7, 8, 9, 10, 11]. Moreover, there are a various of applications of P systems, for instance, ecology and structural biology [12, 13], function optimization [14], machine learning [15, 16, 17], image and signal processing [18, 19, 20, 21, 22, 23].

Spiking neural P systems (SNP systems, in short) were first proposed by Ionescu et al. [6], inspired by the way of transmitting and exchanging information, by means of spikes, between neurons. SNP systems are also a class of distributed and parallel computation systems. Directed graphs are used to express SNP systems, where the nodes are the neurons and the arcs are used to denote the synapses between these neurons. Each SNP system consists of two components: data and rules. The data is denoted by the number of spikes contained in it and is evolved by rules. There are two forms of rules: spiking rule and forgetting rule. Spiking rule has the form $E/a^c \rightarrow a^p$, where E is a regular expression over $\{a\}$, c is the number of spikes consumed by the rule and p is the number of the produced spikes, and $c \geq p \geq 1$. The semantics of spiking rule can be explained as follows. Suppose that neuron σ has a spiking rule $E/a^c \rightarrow a^p$ and contains n spikes satisfying $a^n \in L(E)$. The neuron fires and consumes c spikes, and then it produces p spikes and sends them to all subsequent neurons connected with it. Forgetting rule has the form $a^s \rightarrow \lambda$, where $s \geq 1$. If forgetting rule $a^s \rightarrow \lambda$ is applied in the neuron, then s spikes are removed from it and no spike is produced.

Most SNP systems work in synchronous mode. A global clock is assumed for the synchronization of all neurons, and all neurons in the systems work in parallel, and the rules in each neuron are applied sequentially. When there are more than one rules can be applied in a neuron, one of them must be chosen non-deterministically and applied.

Generally, there are three main research topics:

1. theoretical works,
2. application and
3. simulation systems.

Many variants of SNP systems were proposed, for example, SNP systems with astrocytes [24, 25], SNP systems with anti-spikes [26], SNP systems with weights [27], SNP systems with thresholds [28], SNP systems with rules on synapses [29, 30, 31], SNP systems with multiple channels [32, 33], coupled neural P systems [34], dynamic threshold neural P systems [35], SNP systems with polarizations [36], SNP system with inhibitory rules [37], dendrite P systems [38], nonlinear SNP systems [39], and so on. In addition, several working modes have been investigated, such as asynchronous mode [40], asynchronous mode with local synchronization [41], and sequential mode [42]. Turing universality is one of computational theory of SNP systems. SNP systems and variants can be considered as four devices: number generating/accepting devices, function computing devices as well as language generating devices. In universality investigation, register machines are often regarded as standard model, because it has been proven that register machines can compute/accept any Turing computable number set and obtain a small universal function computing device. Therefore, by simulating register machines, it has been proven that most variants of SNP systems are Turing universal [43, 44]. Moreover, fuzzy logic was introduced into SNP systems to propose a variety of fuzzy spiking neural P systems [45, 46], which have been applied in fault diagnosis [47, 48, 49, 50]. In addition, a number of simulation systems have been developed, for example, P-Lingua [51] and SNP system simulator on GPU [52].

It is no doubt that the synchronization plays a vital role in the proof of above results, however, the assumption of global clock is rather natural from a neurobiological point of view. Therefore, SNP systems working in non-synchronous modes have received much attention in the recent years, especially, in asynchronous mode. The SNP systems working in asynchronous mode are called asynchronous SNP systems (ASNP systems, in short). In asynchronous mode, the global clock is removed and every neuron is not obligatory to use its rules. Therefore, each neuron is free to choose time to fire without any time restriction when its rules are available. New spikes that are received from adjacent neurons may cause the rules to no longer be available. In this case, the computation will continue to work under the new configuration. The result of the computation is no longer relevant with the distance in time because of the asynchronous working mode. Thus, the total number of spikes, which is sent out to the environment, is regarded as the result of the computation. Cavaliere et al. [40] provided a specific description about asynchronization and proved that asynchronous SNP systems with extended rules are equivalent with Turing machines. After that, several asynchronous SNP systems have been investigated, such as asynchronous SNP systems with local synchronization [41], asynchronous SNP systems with rules on synapses [53], asynchronous SNP systems with structural plasticity [54] and asynchronous SNP systems with anti-spikes [55]. These asynchronous systems have been proven to be Turing universal. In addition, Cavaliere et al. [56] investigated the decidability and undecidability of asynchronous SNP systems. The language generating problems of asynchronous SNP systems have been discussed in Zhang et al. [57].

This work discusses a new variant of SNP systems, which works in asynchronous mode and has two interesting features (multiple channels and multiple symbols), asynchronous spiking neural P systems with multiple channels and symbols (ASNP-MCS systems, in short). ASNP-MCS systems are different from the existing asynchronous SNP systems in the following two aspects:

1. Every neuron in ASNP-MCS systems has one or more synaptic channels, thus, different sets of subsequent neurons can be connected with it. It is suitable for ASNP-MCS systems, because of the multiple channels feature, to characterize higher-order dynamic systems.
2. More than one symbols are considered in ASNP-MCS systems. Therefore, the rules in ASNP-MCS systems are extended to handle these multiple symbols. It is also suitable for ASNP-MCS systems, due to the multiple symbols feature, to simulate some complicated systems with different parts.

This work investigates the computational power of ASNP-MCS systems. By simulating register machine, it is proven that ASNP-MCS systems as number generating/accepting devices are Turing universal. In addition, we construct an ASNP-MCS system with 67 neurons as a small universal function computing device. The result can be interpreted as the reason that the loss of computational power caused by removing synchronization can be offset by the use of multiple channels and multiple symbols.

The remainder of this paper is organized as follows. In Section 2, we review some basic mathematical knowledge that will be useful for investigation of universality. The definition of ASNP-MCS systems and an illustrative examples are given in Section 3. In Section 4, we first discuss the computational power of ASNP-MCS systems as number generating/accepting devices, and then we construct a small universal ASNP-MCS system for computing functions. Finally, conclusions and future work are drawn in Section 5.

2 PRELIMINARIES

It is assumed that readers have some knowledge with formal language theory and membrane computing. Basic notions and notations are reviewed in this section.

For an alphabet O, the set of all finite strings of symbols from O is denoted by O^*, and the set of all nonempty strings over O is denoted by O^+; the empty string is denoted by λ.

A regular expression over an alphabet O is defined as follows:

1. λ and each $a \in O$ is a regular expression;
2. if E_1 and E_2 are regular expressions over O, then $(E_1)(E_2)$, $(E_1) \bigcup (E_2)$ and $(E_1)^+$ are regular expressions over O, and
3. nothing else is a regular expression over O.

With each regular expression E we associate with a language $L(E)$, defined in the following way:

1. $L(\lambda) = \{\lambda\}$ and $L(a) = \{a\}$, for all $a \in O$;
2. $L((E_1) \bigcup (E_2)) = L(E_1) \bigcup L(E_2)$, $L((E_1)(E_2)) = L(E_1)L(E_2)$, and $L((E_1)^+) = (L(E_1))^+$, for all regular expressions E_1, E_2 over O.

When writing a regular expression, unnecessary parentheses can be omitted. We can simply write $E^+ \bigcup \{\lambda\}$ as E^*.

A register machine, which is a construct $M = (m, H, l_0, l_h, I)$, can be used to prove the universality of ASNP-MCS systems, where m is the number of registers, H is the set of instruction labels, l_0 is the starting label, l_h is the halting label (assigned to instruction HALT), and I is the set of instructions. Each label from H corresponds to an instruction from I. There are instructions of three forms:

1. l_i : (ADD$(r), l_j, l_k$) (add 1 to register r then go non-deterministically to one of the instructions with labels l_j, l_k).
2. l_i : (SUB$(r), l_j, l_k$) (if register r is non-zero, then subtract 1 from it and go to the instruction with label l_j; otherwise, go to the instruction with label l_k).
3. l_h : HALT (the halting instruction).

It is well-known that a register machine M can generate/accept any Turing computable number set (denoted by NRE).

Number n can be generated by register machine in the following way. The register machine starts with all registers empty (for example, storing the number zero). What instruction activated first is the instruction with label l_0. Then, the subsequent instructions are processed in order. If the register machine reaches the halting instruction, then the computation is completed, and the result of the computation is the number n stored in the first register r_0. We denote by $N_{gen}(M)$ the set of all numbers generated by M.

The register machine M can also accept the numbers. We denote by $N_{acc}(M)$ the set of numbers accepted by M. Its working mechanism can be illustrated as follows. At the beginning, all registers are empty except the first register, and a number is introduced into the first register. In accepting mode, register machine is deterministic, meaning that l_i : (ADD$(r), l_j$) is used to substitute l_i : (ADD$(r), l_j, l_k$) as the ADD instruction.

Functions of form $f : N^k \to N$ can be computed by register machines. It works as follows: at the beginning, all registers are empty and k parameters are introduced into k specific registers; register machine M starts with instruction l_0, and then continues a series of computations until it reaches the halting instruction l_h. The computed function value will be stored in a specific register r when the system halts. In computing mode, M is deterministic, where ADD instructions have the form l_i : (ADD$(r), l_j$).

Korec [58] introduced a small universal register machine, $M_u = (8, H, l_0, l_h, I)$, for computing functions, shown in Figure 1. The register machine contains 8 registers

and 23 instructions. By introducing numbers $g(x)$ and y into registers 1 and 2, respectively, the register machine M_u can compute function $\varphi_x(y) = M_u(g(x), y)$, where g is a recursively function for all natural numbers x and y. When register machine M_u halts, the number stored in the register 0 is the computed function value.

In this work, we will discuss the universality of ASNP-MCS systems as function computing devices by means of register machine M_u as a standard model.

l_0: (*SUB*(1),l_1,l_2)	l_1: (*ADD*(7),l_0)	l_2: (*ADD*(6),l_3)
l_3: (*SUB*(5),l_2,l_4)	l_4: (*SUB*(6),l_5,l_3)	l_5: (*ADD*(5),l_6)
l_6: (*SUB*(7),l_7,l_8)	l_7: (*ADD*(1),l_4)	l_8: (*SUB*(6),l_9,l_0)
l_9: (*ADD*(6),l_{10})	l_{10}: (*SUB*(4),l_0,l_{11})	l_{11}: (*SUB*(5),l_{12},l_{13})
l_{12}: (*SUB*(5),l_{14},l_{15})	l_{13}: (*SUB*(2),l_{18},l_{19})	l_{14}:(*SUB*(5),l_{16},l_{17})
l_{15}: (*SUB*(3),l_{18},l_{20})	l_{16}: (*ADD*(4),l_{11})	l_{17}: (*ADD*(2),l_{21})
l_{18}: (*SUB*(4),l_0,l_h)	l_{19}: (*SUB*(0),l_0,l_{18})	l_{20}: (*ADD*(0),l_0)
l_{21}: (*ADD*(3),l_{18})	l_h:*HALT*	

Figure 1. A small universal register machine M_u

3 ASYNCHRONOUS SPIKING NEURAL P SYSTEMS WITH MULTIPLE CHANNELS AND SYMBOLS

3.1 Definition

Definition 1. An ASNP-MCS system, of degree $m \geq 1$, is a construct:

$$\Pi = (O, L, \sigma_1, \sigma_2, \ldots, \sigma_m, \text{syn}, \text{in}, \text{out})$$

where

1. $O = \{a_1, a_2, \ldots, a_k\}$ is the alphabet ($a_1, a_2, \ldots, a_k$ denote k types of spikes, respectively);
2. $L = \{1, 2, \ldots, N\}$ is the alphabet of channel labels;
3. $\sigma_1, \ldots, \sigma_m$ are neurons, of the form $\sigma_i = (\vec{n}_i, L_i, R_i)$, $1 \leq i \leq m$, where
   (a) $\vec{n}_i = (n_{i1}, n_{i2}, \ldots, n_{ik})$ is a k-dimensional vector, where $n_{ij} \geq 0$ is the initial number of spikes of jth type a_j contained in neuron σ_i, $1 \leq j \leq k$;
   (b) $L_i \subseteq L$ is a finite set of channel labels used in neuron σ_i;
   (c) R_i is a finite set of rules of the following two forms:
      i Spiking rule $E/a_1^{c_1}a_2^{c_2}\cdots a_k^{c_k} \rightarrow a_1^{p_1}a_2^{p_2}\cdots a_k^{p_k}(l)$, where E is a regular expression over O, and $c_j \geq 0$, $p_j \geq 0$ $(1 \leq j \leq k)$, $c_1 + c_2 + \cdots + c_k \geq p_1 + p_2 + \cdots + p_k \geq 1$, $l \in L_i$;
      ii Forgetting rule $a_1^{s_1}a_2^{s_2}\cdots a_k^{s_k} \rightarrow \lambda$, where $s_j \geq 0$ $(1 \leq j \leq k)$ and $s_1 + s_2 + \cdots + s_k \geq 1$, with the restriction that for each rule $E/a_1^{c_1}a_2^{c_2}\cdots a_k^{c_k} \rightarrow a_1^{p_1}a_2^{p_2}\cdots a_k^{p_k}(l)$ of type (i) from R_i, we have $a_1^{s_1}a_2^{s_2}\cdots a_k^{s_k} \notin L(E)$;

4. syn $= \{(i,j,l)\} \subseteq \{1,2,\ldots,m\} \times \{1,2,\ldots,m\} \times L$ with $(i,i,l) \notin$ syn for $\forall 1 \leq i \leq m$ and $\forall l \in L$ (synapse connections);
5. in indicates the input neuron of the system;
6. out indicates the output neuron of the system.

There is only one single type of spikes, denoted by symbol a, in SNP systems, while ASNP-MCS systems proposed in this work have several distinct types of spikes, denoted by symbols a_1, a_2, ..., a_k, and these multiple symbols can be interpreted as electrical signals with distinct frequencies. The number of spikes in every neuron in an SNP system is denoted by a natural number. However, since there are spikes of k types in an ASNP-MCS system, a k-dimensional vector, $\vec{n}_i = (n_{i1}, n_{i2}, \ldots, n_{ik})$, is considered to indicate the number of each neuron. If $n_{ij} > 0$, one or more spikes of type a_j are contained in the neuron σ_i. If $n_{ij} = 0$, there is no spike of type a_j in the neuron σ_i.

An ASNP-MCS system can be represented by a directed graph, where the nodes are used for labeling m neurons and the arcs are used for denoting the synapses between these neurons. The connection relationships between m neurons are described by syn $= \{(i,j,l)\} \subseteq \{1,2,\ldots,m\} \times \{1,2,\ldots,m\} \times L$, meaning that neuron σ_i connects neuron σ_j via channel (l) .

Since ASNP-MCS systems work in asynchronous mode, the use of rules in neurons is not obligatory, which means that a rule can be used immediately or also can be used later if the spikes in the neuron enable the rule. If the number of spikes in the neuron has been changed before using the rule, such as new spikes coming, and the present rule can not be used any more, then the computation continues in the new circumstance. Notice that the produced spikes in the neuron will be sent immediately when the neuron applies the rule in a later step.

Spiking rules and forgetting rules also exist in ASNP-MCS systems. What is the meaning of spiking rules of form (i) can be explained as follows. The rule $E/a_1^{c_1}a_2^{c_2}\cdots a_k^{c_k} \rightarrow a_1^{p_1}a_2^{p_2}\cdots a_k^{p_k}(l)$ in neuron σ_i can be applied at some time with the condition of $a_1^{n_1}a_2^{n_2}\cdots a_k^{n_k} \in L(E)$. When neuron σ_i fires, c_j spikes of type a_j are consumed (thus $n_j - c_j$ spikes of type a_j are remained), and p_j spikes of type a_j are produced, $1 \leq j \leq k$. The spikes generated by neuron σ_i are sent to the subsequent neurons via channel (l). The semantics of forgetting rules of form (ii) can be described as follows. If the spikes in neuron σ_i are exactly s_j spikes of type a_j, $1 \leq j \leq k$, then the rule $a_1^{s_1}a_2^{s_2}\cdots a_k^{s_k} \rightarrow \lambda$ can be enabled, which means that all s_j spikes of type a_j are removed from neuron σ_i.

We use the following $m \times k$ matrix to describe the initial configuration in ASNP-MCS systems,

$$C_0 = \begin{pmatrix} n_{11} & n_{12} & \cdots & n_{1k} \\ n_{21} & n_{22} & \cdots & n_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ n_{m1} & n_{m2} & \cdots & n_{mk} \end{pmatrix}_{m \times k}$$

where n_{ij} is the initial number of spikes of type a_j contained in neuron σ_i, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, k$. Besides, the similar $m \times k$ matrix also can describe the configuration at any time t during the computation. The transitions among configurations can be therefore defined. One can define a computation with the sequence of transitions starting from the initial configuration. When there is no rule which can be applied and no "block" condition happened, the computation halts.

In SNP systems, we can associate any computation (halting or not) with a spike train: the sequence of zeros and ones describing the behavior whether the output neuron spikes. Since ASNP-MCS systems work in asynchronous mode, zeros sent out by the output neuron are normal. Thus, we define the result of a computation as the number of ones sent out by the output neuron starting from the initial configuration. We denote by $N_{gen}(\Pi)$ the set of numbers generated by Π, where subscript *gen* indicates that the present system works in generating mode. We denote by $N_{gen}\text{ASNP-MCS}_m^n$ the family of all sets $N_{gen}(\Pi)$ computed by ASNP-MCS systems with at most m neurons and at most n rules in every neuron.

An ASNP-MCS system Π can work in accepting mode. There is no output neuron any more. Instead, the input neuron is added to receive a spike train from the environment. The system Π begins the computation by reading a spike train from the environment, and $3\text{T}n$ spikes of type a is stored in a specific neuron. Denote by $N_{acc}(\Pi)$ the set of numbers accepted by system Π, where subscript *acc* represents that the system works in accepting mode. Denote by $N_{acc}\text{ASNP-MCS}_m^n$ the family of all sets $N_{acc}(\Pi)$ accepted by ASNP-MCS systems, which have at most m neurons and at most n rules in every neuron.

3.2 An Example

An example is provided to explain the differences between an SNP-MCS system and an ASNP-MCS system, shown in Figure 2. SNP-MCS system works in synchronous mode, while ASNP-MCS system works in asynchronous mode. There are three neurons, σ_1, σ_2 and σ_3, labeled by 1, 2 and 3 in Figure 2. Neuron σ_1 has the spikes of two types (type a and type b), and it has only one synaptic channel labeled by (1). Neuron σ_3 contains the spikes of two types, however, it has two different synaptic channels, labeled by (1) and (2). There is only one type of spikes, type a, and a single synaptic channel labeled by (1) in neuron σ_2. Neuron σ_1 starts with a spike of type a, and neuron σ_3 starts with a spike of type b. What the mainly difference between an SNP-MCS system and an ASNP-MCS system is their working mode.

In the synchronous mode, since neuron σ_1 initially has a spike of type a and neuron σ_3 initially has a spike of type b, the spikes enable rule $a \to b(1)$ and rule $b \to a(1)$ in neurons σ_1 and σ_3, respectively. Thus, two rules can be applied simultaneously. Neuron σ_1 sends a spike of type b to neuron σ_3 via channel (1). Neuron σ_3 emits a spike of type a to neuron σ_1 through channel (1). When neuron σ_1 receives a spike of type a from neuron σ_3, it can apply rule $a \to b(1)$ again. Similarly, neuron σ_3 also receives a spike of type b from neuron σ_1 and it can apply its rule $b \to a(1)$, too. Therefore, the two neurons simultaneously fire again and

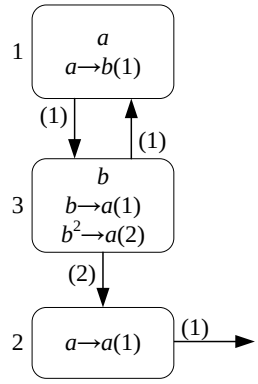


Figure 2. An example of spiking neural P systems with multiple channels and symbols

exchange a spike with each other. Repeatedly, in every step, neurons σ_1 and σ_3 will exchange a spike, and no spike is sent to the environment. Therefore, there is no result of the computation.

For the ASNP-MCS system that works in the asynchronous mode, when the number of spikes in a neuron enables its rule, the neuron can fire its rule at some time, sooner or later. Here, the rules in neurons σ_1 and σ_3 are available separately, but neurons σ_1 and σ_3 are free to choose a time to fire. Thus, there are three cases.

Case 1: Neuron σ_1 fires before neuron σ_3. At first, rule $a \to b(1)$ in neuron σ_1 can be applied, which means that neuron σ_1 sends a spike of type b to neuron σ_3. The number of spikes in neuron σ_3 has been changed to two spikes of type b, after receiving a spike sent by neuron σ_1. Thus, rule $b^2 \to a(1)$ can be applied. At a later time, neuron σ_3 fires and sends a spike of type a to neuron σ_2 through channel (2). Rule $a \to a(1)$ in neuron σ_2 can be applied when it receives a spike, and neuron σ_2 sends a spike of type a to the environment. In this case, only one single spike is sent to the environment. Therefore, the computation result of the system is 1.

Case 2: Neuron σ_1 fires later than neuron σ_3. Since there is a spike of type b in neuron σ_3, rule $b \to a(1)$ can be applied, and neuron σ_1 receives a spike of type a from neuron σ_3 via channel (1). The number of spikes in neuron σ_1 are two now and there is no rule can be applied, thus the computation is blocked. We can know that there is no any output, thus no any computation result.

Case 3: Neurons σ_1 and σ_3 fire together. In this case, the running of an ASNP-MCS system is the same with the situation in synchronous mode at the first round. So, after the first round, the number of spikes in every neuron is back to its available state and no spike is sent out. Then, neurons σ_1 and σ_3 face a choice again: who want to fire first? And there are also three choices:

1. If it is the Case 1, neuron σ_1 fires before neuron σ_3, then a spike will be sent from the system and the computation will halt. Hence, the final result of the computation is 1;
2. If it is the Case 2, neuron σ_1 fires later than neuron σ_3, then there will be no result in the second round and the computation will be blocked;
3. If it is the Case 3, which means that the two neurons fire together, no spike will be sent to the environment and the numbers of spikes in neurons σ_1 and σ_3 will be back to their available state again.

And then there will be also the three choices again at the next round. The above situation is repeated again. Therefore, if the neurons choose Case 3 at the second round, the final computation result of the system will be one of the following three results: generating the result (number 1), blocked and infinitive repeat.

From the description above, we know that ASNP-MCS systems, which work in the asynchronous mode, have more nondeterministic results compared with SNP-MCS systems.

4 UNIVERSALITY RESULTS

In this section, we will discuss the Turing universality of asynchronous spiking neural P systems with multiple channels and symbols (ASNP-MCS systems, in short) as number generating/accepting devices and function computing devices. We prove the universality by simulating the register machine. The systems proposed can generate/accept any sets of recursively enumerable numbers (the family of sets of recursively enumerable numbers is denoted by NRE) and any recursively enumerable computational function.

We construct an ASNP-MCS system Π to simulate register machine M and the result computed by M is presented by the number of spikes that the output neuron sends to the environment. Without loss of generality, two symbols, a and b, are considered in Π, thus $O = \{a, b\}$. INPUT module, ADD module, SUB module and FIN module are constructed to simulate instructions of M, shown in Figures 3, 4, 5, 6, 7 and 8. If t_r is the number of all SUB instructions that act on the same register r, a constant is defined as follows:

$$T = 8 * \max\{t_r | 0 \leq r \leq 7\} = 8 * t_5 = 8 * 4 = 32.$$

The following rule is used to activate instruction neurons: when instruction neurons σ_{l_i}, σ_{l_j} and σ_{l_k} receive $3T$ spikes of type b, they will be activated and execute the corresponding operations. In the process of the computation, the content of register r is coded by the number of spikes in neuron σ_r through the following way: if there is a number $n(\geq 0)$ in register r, then neuron σ_r has $3\mathrm{T}n$ spikes of type a, vice versa.

Rule $b^{3T} \to a^{\psi(p)}(q)$ is a formal spiking expression, used in INPUT, ADD, SUB and FIN modules. It works as follows. When the rule is available, the neuron consumes 3T spikes of type b and sends $\psi(p)$ spikes of type a to the subsequent neurons connected with the neuron, where the rule uses channel (q). We define the function $\psi(p)$ on the sets of instruction symbols as follows:

$$\psi(p) = \begin{cases} 3T, & \text{if } p \text{ is an ADD instruction;} \\ 2T + s, & \text{if } p \text{ is } i^{\text{th}} \text{ SUB instruction in all SUB instructions on register } r; \\ 1, & \text{if } p \text{ is the output instruction.} \end{cases}$$

In this paper, we use the following way to define the value of s in the SUB instructions which act on the same register r:

- If the instruction is the first SUB instruction in register r, then $s = 1$;
- If the instruction is the second SUB instruction in register r, then $s = 2$;
- If the instruction is the third one in register r, then $s = 4$;
- If the instruction is the forth one, then $s = 9$.

For example, there are four SUB instructions acted on register 5 in universal register machine M_u, $l_3 : (\text{SUB}(5), l_2, l_4)$, $l_{11} : (\text{SUB}(5), l_{12}, l_{13})$, $l_{12} : (\text{SUB}(5), l_{14}, l_{15})$, $l_{14} : (\text{SUB}(5), l_{16}, l_{17})$. Therefore, because the first SUB instruction in register 5 is instruction $l_3 : (\text{SUB}(5), l_2, l_4)$, the function is $\psi(p) = 2T + 1$. The second SUB instruction is instruction l_{11} and its function is $\psi(p) = 2T + 2$. Similarly, the function of the third SUB instruction, $l_{12} : (\text{SUB}(5), l_{14}, l_{15})$, is $\psi(p) = 2T + 4$. The function of the forth SUB instruction l_{14} is $\psi(p) = 2T + 9$.

4.1 ASNP-MCS Systems as Number Generating Devices

Theorem 4.1. $N_{gen}\text{ASNP-MCS}_*^3 = \text{NRE}$.

Proof. It is only proven that $\text{NRE} \subseteq N_{gen}\text{ASNP-MCS}_*^3$, because it is obvious for conclusion $N_{gen}\text{ASNP-MCS}_*^3 \subseteq \text{NRE}$. To this aim, we characterize NRE by a non-deterministic register machine M working in generating mode.

In order to prove this conclusion, we construct an ASNP-MCS system Π_1 to simulate the register machine M. The system consists of three parts, ADD module, SUB module and FIN module, which simulate ADD instruction, SUB instruction and halting instruction, respectively.

1. ADD module, simulating $l_i : (\text{ADD}(r), l_j, l_k)$.

   The ADD module is shown in Figure 3. Suppose that we simulate instruction $l_i : (\text{ADD}(r), l_j, l_k)$ at some time, which means that neuron σ_{l_i} has 3T spikes of type b and no any spikes are in other neurons except those associated with registers. The rule $b^{3T} \to a^{3T}(1)$ in neuron σ_{li} is applied at some time and then neuron σ_{li} sends 3T spikes of type a to neurons σ_r and $\sigma_{l_{i1}}$. When neuron σ_r

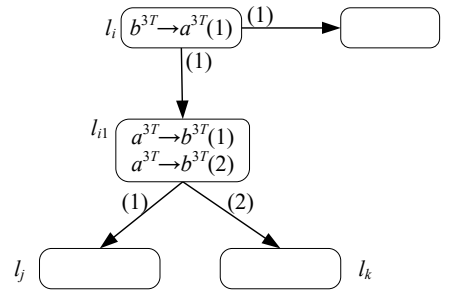


Figure 3. A nondeterministic ADD module

receives the spikes from neuron σ_{l_i}, number 1 is added into the corresponding register. Two rules in neuron $\sigma_{l_{i1}}$ become available after it receives the spikes from neuron σ_{l_i}, but one of the two rules can be applied non-deterministically. If rule $a^{3T} \to b^{3T}(1)$ is applied, neuron σ_{l_j} will receive 3T spikes of type b via channel (1). If rule $a^{3T} \to b^{3T}(2)$ is applied, neuron $\sigma_{l_{i1}}$ will send the spikes to neuron σ_{l_k} through channel (2).

As can be seen from above, ADD module can correctly simulate ADD instruction: starting from $3T$ spikes of type b in neuron σ_{l_i}, $3T$ spikes of type a are added in neuron σ_r (meaning that the corresponding register r is added by 1), and one of two instruction neurons, σ_{l_j} and σ_{l_k}, receives $3T$ spikes of type b non-deterministically.

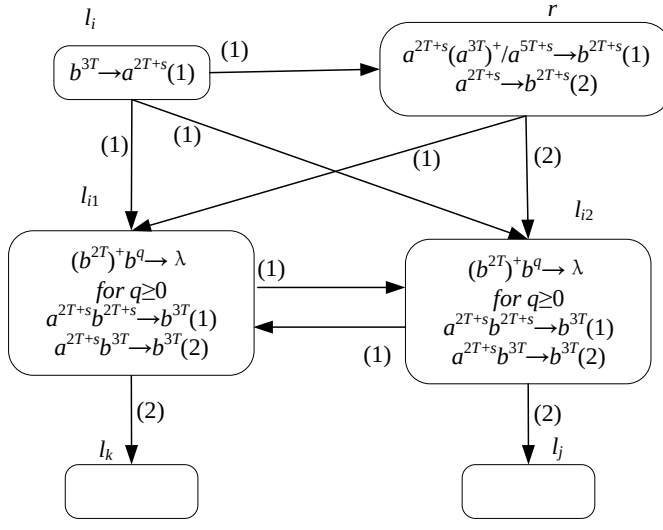


Figure 4. SUB module

2. SUB module, simulating $l_i : (\text{SUB}(r), l_j, l_k)$.

   This module is shown in Figure 4. Suppose that a SUB instruction is simulated at some time, 3T spikes of type b are in neuron σ_{l_i} and no spike is in other neurons except neuron σ_r (i.e., the multiple of 3T^{th} spikes in neuron σ_r is the number in the corresponding register r). The SUB instruction works as follows. At some time, neuronσ_{l_i} fires, and then 3T spikes of type b are consumed and $2T+s$ spikes of type a are produced, and the generated spikes are sent to neurons σ_r, $\sigma_{l_{i1}}$ and $\sigma_{l_{i2}}$ via channel (1). Neuron σ_r fires, at a later time, according to the number of spikes in it:

   (a) If there are several spikes in neuron σ_r, indicating that the number in register r is not zero, then rule $a^{2T+s}(a^{3T})^+/a^{5T+s} \to b^{2T+s}(1)$ can be applied;
   (b) If there is no spike in neuron σ_r, indicating that the number in register r is 0, then rule $a^{2T+s} \to b^{2T+s}(2)$ can be applied. There are the following two cases.

   Case 1: If several spikes are in neuron σ_r (i.e., the number in register r is greater than 0), then, at a later time, rule $a^{2T+s}(a^{3T})^+/a^{5T+s} \to b^{2T+s}(1)$ can be applied. Therefore, neuron σ_r consumes $5T+s$ spikes of type a and produces $2T+s$ spikes of type b, and then it sends the produced spikes to neuron $\sigma_{l_{i1}}$ by channel (1). As a result, neuron $\sigma_{l_{i1}}$ has $2T+s$ spikes of type a sent by neuron σ_{l_i} and $2T+s$ spikes of type b sent by neuron σ_r. Hence, rule $a^{2T+s}b^{2T+s} \to b^{3T}(1)$ can be applied. At a later time, neuron $\sigma_{l_{i1}}$ fires and transmits 3T spikes of type b to neuron $\sigma_{l_{i2}}$. Neuron $\sigma_{l_{i2}}$ receives $2T+s$ spikes of type a sent by neuron σ_{l_i} and 3T spikes of type b sent by neuron $\sigma_{l_{i1}}$. So, rule $a^{2T+s}b^{3T} \to b^{3T}(2)$ is enabled. At some time, neuron $\sigma_{l_{i2}}$ fires and emits 3T spikes of type b to neuron σ_{l_j}.

   Case 2: If no spike is in neuron σ_r, indicating that the number in register r is 0, then, at a later time, rule $a^{2T+s} \to b^{2T+s}(2)$ is applied to send $2T+s$ spikes of type b to neuron $\sigma_{l_{i2}}$ via channel (2). Except from receiving $2T+s$ spikes of type b, neuron $\sigma_{l_{i2}}$ also receives $2T+s$ spikes of type a from neuron σ_{l_i}. Therefore, the spikes in the neuron enable rule $a^{2T+s}b^{2T+s} \to b^{3T}(1)$. Then, at some time, the rule in neuron $\sigma_{l_{i2}}$ is applied to send 3T spikes of type b to neuron $\sigma_{l_{i1}}$. With $2T+s$ spikes of type a and 3T spikes of type b in neuron $\sigma_{l_{i1}}$, rule $a^{2T+s}b^{3T} \to b^{3T}(2)$ can be used at a later time. Therefore, neuron $\sigma_{l_{i1}}$ sends 3T spikes of type b to neuron σ_{l_k} via channel (2).

   Note that there is interference between SUB modules and other modules, which means that the same register could be operated by different instructions. Here an example is provided to illustrate this question: instructions $l_{11} : (\text{SUB}(5), l_{12}, l_{13})$ and $l_{12} : (\text{SUB}(5), l_{14}, l_{15})$, for instance, all act on the register 5. The SUB modules that act on the same register, register 5 here, all will receive the spikes via channel (1) (the number in register is not null) or via channel (2) (the number in register is 0) from neuron σ_r. Specifically, suppose that instruction l_{11}

is activated, so the corresponding neurons in SUB module of instruction l_{11} is activated (active neurons, in short), too. Other instructions, like l_{12} here, are non-activated, so the corresponding neurons in SUB modules of other instructions, no instruction l_{11}, are also not activated (passive neurons for short). Instruction l_{11} is the second SUB instruction of register 5. Therefore, the function of its sets of instruction symbols is $\psi(p) = 2T + s = 2T + 2$. So, neuron σ_r will send $2T + 2$ spikes of type b to the following neurons which are also connected with register 5 through channel (1) if register 5 contains numbers. This means, not only active neuron $\sigma_{l_{111}}$ (neuron $\sigma_{l_{i1}}$ connects with neuron σ_r via channel (1) in the module of l_{11}) but also passive neuron $\sigma_{l_{121}}$ (neuron $\sigma_{l_{i1}}$ connects with neuron σ_r via channel (1) in the module of l_{12}) as well as other passive neurons that are connected with neuron σ_r via channel (1) can receive the spikes. For active neurons, the spikes received are exactly they want. However, the passive neurons receive the spikes in a wrong way and cannot refuse to accept them because of the interference between neurons, so we called these spikes, "wrong spikes".

We can find that, if neurons $\sigma_{l_{i1}}$ or $\sigma_{l_{i2}}$ receive the "wrong spikes" before the instruction is activated, then the rule $(b^{2T})^+ \to b^q$, $q \geq 0$, can be applied, and neurons $\sigma_{l_{i1}}$ or $\sigma_{l_{i2}}$ will remove the "wrong spikes". There is also a possible situation that there still are "wrong spikes" until the instruction is activated. Therefore, when SUB module of active instruction starts running, neuron σ_{l_i} emits the spikes to neurons $\sigma_{l_{i1}}$ and $\sigma_{l_{i2}}$, and no rule in the two neurons can be used because of the "wrong spikes". If the "wrong spikes", new or not, cause that no rule can be applied in neurons $\sigma_{l_{i1}}$ or $\sigma_{l_{i2}}$, then the computation is blocked. Since the system works asynchronously, another "bad" situation must be considered. Specifically, before the new computation starts, there are still several "wrong spikes" that are not removed in last computation. In this case, the new computation will be blocked, too, when neuron σ_{l_i} sends the spikes by its rule. When the computation is blocked, no spike is sent out. No error results appear in this situation and the output is the computation result when the system finally reaches instruction l_h. Therefore, the system correctly simulates the SUB instruction of register machine M.

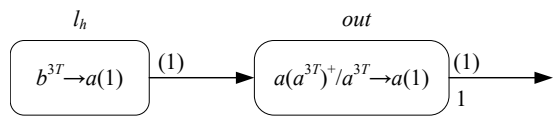


Figure 5. FIN module

3. FIN module, outputting the computation result.

Figure 5 shows this module. Suppose that the computation of M stops at some time, meaning that M receives the halting instruction. For system Π_1, this indicates neuron σ_{l_h} receives 3T spikes of type b. Thus, rule $b^{3T} \to a(1)$ in

neuron σ_{l_h} can be applied at a later time, and then a spike of type a is sent to the output neuron by channel (1). After receiving the spike from neuron σ_{l_h}, neuron σ_{out} transmits a spike of type a to the environment constantly when it fires continually, until the moment no rule can be applied in the output neuron. The number of spikes sent to the environment is the results of the computation.

From the description of working modules in system Π_1, we know that the register machine M can be correctly simulated by system Π_1 with at most two kinds of spikes and at most three rules in each neuron. Therefore, the theorem holds. □

4.2 ASNP-MCS Systems as Number Accepting Devices

Theorem 4.2. $N_{acc}\text{ASNP-MCS}_*^3 = \text{NRE}$.

Proof. Like the proof of Theorem 4.1, we construct an ASNP-MCS system Π_2 to simulate a deterministic register machine M, $M = (m, H, l_0, l_h, I)$. There are three parts in system Π_2: INPUT module, deterministic ADD module and SUB module. The INPUT module is used to input a spike train from the environment, shown in Figure 6. Spike train $a^{2T}(a^{3T})^n b^T$ can be read by neuron σ_{in} from the environment, and then $3Tn$ spikes of type a are stored in neuron σ_1, where the multiple of $3T$ spikes of type a is n, indicating that the accepted number is n. At some time, neuron σ_{in} reads $2T$ spikes of type a and several groups of $3T$ spikes of type a from the environment. Therefore, rule $a^{2T}(a^{3T})^+/a^{3T} \to a^{3T}(1)$ can be applied from reading the first $3T$ spikes of type a to receiving last ones. As a result, neuron σ_{in} sends $3T$ spikes of type a to neuron σ_1 via channel (1) once the rule is applied. When neuron σ_1 receives $3T$ spikes of type a from neuron σ_{in}, the number in register is added by 1. There are two cases before neuron σ_{in} reads T spikes of type b.

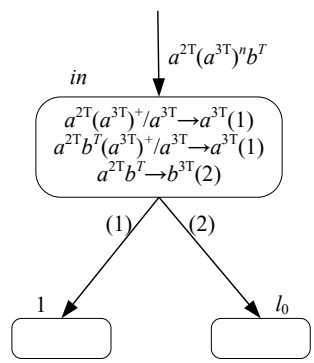


Figure 6. INPUT module

Case 1: Though $3Tn$ spikes of type a have been read from the environment, there are still several groups of $3T$ spikes of type a in neuron σ_{in} because of the

asynchronous mode. After T spikes of type b in spike train are received, rule $a^{2T}(a^{3T})^+b^T/a^{3T} \rightarrow a^{3T}(1)$ becomes available. Since neuron σ_{in} has read all spikes in the spike train, no new spike comes. Thus, the rule can be applied at a later time and neuron σ_{in} will send the spikes to neuron σ_1 every time as the rule is applied. When all $3Tn$ spikes of type a are stored in neuron σ_1, turn to Case 2.

Case 2: $3Tn$ spikes of type a are all stored in neuron σ_1. When T spikes of type b are read or when T spikes of type b have already been in neuron σ_1 (the situation that Case 1 turns to Case 2), there are only $2T$ spikes of type a and T spikes of type b left in neuron σ_{in}. Hence, rule $a^{2T}b^T \rightarrow b^{3T}(2)$ can be applied at some time, and $3T$ spikes of type b will be sent to neuron σ_{l_0} via channel (2).

Therefore, neuron σ_1 receives $3Tn$ spikes of type a, which means that the number stored in register 1 is n. Besides, since neuron σ_{l_0} receives 3T spikes of type b, the system starts to simulate the initial instruction l_0 of M.

We also can get something more. Neuron σ_1 will receive $3Tn$ spikes of type a and neuron σ_{l_0} will receive $3T$ spikes of type b after neuron σ_{in} reads the spike train, $a^{2T}(a^{3T})^+b^T$, from the environment, which means the result of the computation can be gotten normally without "block" situation.

When a register machine works in accepting mode, its ADD instruction of form $l_i : (\mathrm{ADD}(r), l_j)$ is deterministic. The deterministic ADD module is shown in Figure 7. Since $3T$ spikes of type b are in neuron σ_{l_i}, then the neuron fires at a later time and transmits $3T$ spikes of type a to neurons $\sigma_{l_{i1}}$ and σ_r via channel (1). The number of spikes in neuron σ_r is added by $3T$, which indicates that the number of the corresponding register is added by 1. After receiving $3T$ spikes of type a from neuron σ_{l_i}, neuron $\sigma_{l_{i1}}$ applies the rule, at a later time, to converse $3T$ spikes of type a to $3T$ spikes of type b and sends the generated spikes to neuron σ_{l_j}. With $3T$ spikes of type b in neuron σ_{l_j}, the system starts to simulate the instruction l_j of M.

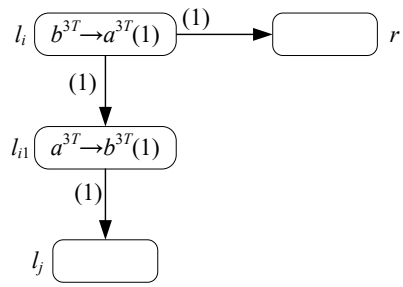


Figure 7. A deterministic ADD module

SUB module remains unchanged, shown in Figure 4. FIN module is removed, but the system remains neuron σ_{l_h}. Since neuron σ_{l_h} has $3T$ spikes of type b, the computation of M reaches instruction l_h and halts.

Based on the discussion above, an ASNP-MCS system with at most three rules in each neuron can correctly simulate the register machine working in accepting mode and no "block" happens in INPUT module. Therefore, the theorem holds. □

4.3 ASNP-MCS Systems as Small Universal Function Computing Devices

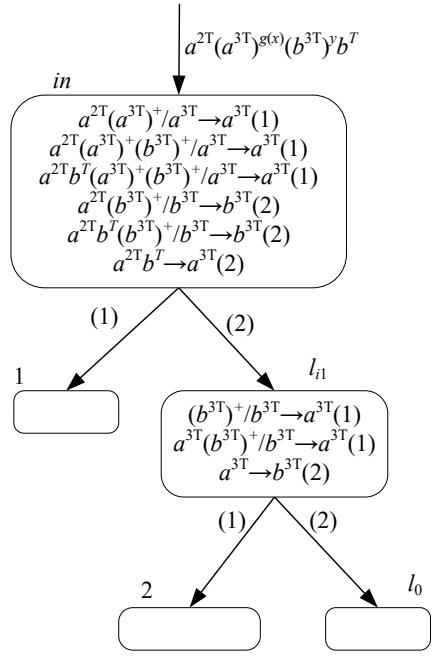


Figure 8. INPUT module

In this section, we will investigate the Turing universality of ASNP-MCS systems as function computing devices. The universality of ASNP-MCS systems will be proved by simulating a small universal register machine M_u.

Theorem 4.3. There is a small universal ASNP-MCS system with 67 neurons for computing functions.

Proof. We construct an ASNP-MCS system Π_3, including INPUT module, ADD module, SUB module and FIN module, to simulate a small universal register machine M_u. The ADD module is the same with ADD module working in accepting mode, shown in Figure 7. The SUB module is the same with SUB module that works in accepting mode, shown in Figure 4. The FIN module is the same with FIN module working in accepting mode, shown in Figure 5. However, the INPUT module is different from INPUT module that works in accepting mode.

The INPUT module, shown in Figure 8, is used for reading a spike train $a^{2T}(a^{3T})^{g(x)}(b^{3T})^{y}b^{T}$ from the environment and introducing $3Tg(x)$ spikes of type a to neuron σ_1 and $3Ty$ spikes of type a to neuron σ_2. At some moment, neuron σ_{in} reads 2T spikes of type a and several groups of $3T$ spikes of type a from the environment. Thus, rule $a^{2T}(a^{3T})^{+}/a^{3T} \to a^{3T}(1)$ can be applied from first $3T$ spikes of type a to last ones coming. Notice that neuron σ_{in} is free to choose to fire or not once the rule is activated by the spikes in neuron σ_{in}. If neuron σ_{in} fires, then $3T$ spikes of type a are produced and the produced spikes are sent to neuron σ_1 by channel (1). If not, neuron σ_{in} goes on reading the next spike in the spike train. Before the first $3T$ spikes of type b come into neuron σ_{in}, indicating that $3Tg(x)$ spikes of type a already have been read, rule $a^{2T}(a^{3T})^{+}/a^{3T} \to a^{3T}(1)$ in neuron σ_{in} can be:

1. No execution, which means that $3Tg(x)$ spikes of type a are still remained in neuron σ_{in};
2. Partial execution, meaning that some groups of $3T$ spikes of type a are still in neuron σ_{in};
3. Total execution (no 3T spikes of type a in neuron σ_{in}). Hence, there are two cases before the first $3T$ spikes of type b are received: remaining groups of $3T$ spikes of type a or not.

The two cases are considered as follows.

Case 1: All the $3Tg(x)$ spikes of type a are stored in neuron σ_1 by the execution of the rule and there are only $2T$ spikes of type a left in neuron σ_{in}. At the period of time when neuron σ_{in} reads the following groups of $3T$ spikes of type b, rule $a^{2T}(b^{3T})^{+}/b^{3T} \to b^{3T}(2)$ is enabled. Thus, neuron σ_{in} fires, at some time, and then it sends $3T$ spikes of type b to neuron $\sigma_{l_{i1}}$. There are two situations when neuron σ_{in} has read all the spikes in the spike train. The two cases are as follows.

1. All the $3Ty$ spikes of type b are sent to neuron $\sigma_{l_{i1}}$ by the rule. There are 2T spikes of type a and T spikes of type b left in neuron σ_{in}. Therefore, rule $a^{2T}b^{T} \to a^{3T}(2)$ can be applied at a later time. Then, neuron σ_{in} transmits $3T$ spikes of type a to neuron $\sigma_{l_{i1}}$ via channel (2).
2. Several groups of $3T$ spikes of type b are remained in neuron σ_{in}. The spikes in neuron σ_{in} enable rule $a^{2T}b^{T}(b^{3T})^{+}/b^{3T} \to b^{3T}(2)$. Thus, the rule can be applied whenever the rule is available. After all the $3Ty$ spikes of type b are sent to neuron $\sigma_{l_{i1}}$, rule $a^{2T}b^{T} \to a^{3T}(2)$ in neuron σ_{in} can be enabled. Finally, neuron σ_{in} sends $3T$ spikes of type a to neuron σ_{li1} through channel (2).

Case 2: There are still some groups of 3T spikes of type a in neuron σ_{in}. After neuron σ_{in} reads the next groups of $3T$ spikes of type b, rule $a^{2T}(a^{3T})^{+}(b^{3T})^{+}/a^{3T} \to a^{3T}(1)$ can be applied. Then, neuron σ_{in} fires at some time, sending $3T$ spikes of

type a to neuron σ_1 through channel (1). At the period of time from reading the first $3T$ spikes of type b to last ones coming, rule $a^{2T}(a^{3T})^+(b^{3T})^+/a^{3T} \to a^{3T}(1)$ or rule $a^{2T}(b^{3T})^+/b^{3T} \to b^{3T}(2)$ can be applied according to the number of spikes in the neuron. If all $3Tg(x)$ spikes of type a are stored in neuron σ_1, rule $a^{2T}(b^{3T})^+/b^{3T} \to b^{3T}(2)$ can be enabled. If not, rule $a^{2T}(a^{3T})^+(b^{3T})^+/a^{3T} \to a^{3T}(1)$ can be applied. There are also three situations after T spikes of type b are read. The three cases are as follows.

1. If all the $3Tg(x)$ spikes of type a are stored in neuron σ_1 and $3Ty$ spikes of type b are sent to neuron $\sigma_{l_{i1}}$, then turn to Case 1 (1).
2. If all the $3Tg(x)$ spikes of type a are stored in neuron σ_1 and $3Ty$ spikes of type b are not sent to neuron $\sigma_{l_{i1}}$, then turn to Case 1 (2).
3. If there are still several groups of 3T spikes of type a in neuron σ_{in}, then rule $a^{2T}b^T(a^{3T})^+(b^{3T})^+/a^{3T} \to a^{3T}(1)$ can be used until no groups of $3T$ spikes of type a are left. Then, turn to Case 1 (2).

After all the $3Tg(x)$ spikes of type a are stored into neuron σ_1, $3Ty$ spikes of type b also will be sent to neuron $\sigma_{l_{i1}}$ sequentially. Thus, rule $(b^{3T})^+/b^{3T} \to a^{3T}(1)$ in neuron $\sigma_{l_{i1}}$ can be applied between the period of time that several groups of $3T$ spikes of type b are received. Neuron $\sigma_{l_{i1}}$ fires at some time, consuming $3T$ spikes of type b, producing $3T$ spikes of type a and transmitting the produced spikes to neuron σ_2. Notice that neuron $\sigma_{l_{i1}}$ is free to fire or not. There are two cases according to the number of spikes in neuron σ_{li1} after $3T$ spikes of type a are received by the neuron.

1. All the $3Ty$ spikes of type b in neuron $\sigma_{l_{i1}}$ are processed by rule $(b^{3T})^+/b^{3T} \to a^{3T}(1)$, which means that neuron σ_2 received $3Ty$ spikes of type a (the corresponding number in register 2 is y). Then, rule $a^{3T} \to b^{3T}(2)$ can be applied, at a later time, and neuron $\sigma_{l_{i1}}$ sends $3T$ spikes of type b to neuron σ_{l_0} via channel (2).
2. There are still some groups of $3T$ spikes of type b in neuron $\sigma_{l_{i1}}$. Thus, rule $a^{3T}(b^{3T})^+/b^{3T} \to a^{3T}(1)$ can be used at a later time. Turn to (1) unless all the $3Ty$ spikes of type b are changed as $3Ty$ spikes of type a and stored in neuron σ_2.

When neuron σ_{l0} receives 3T spikes of type b, the system starts to simulate initial instruction σ_{l_0} of M.

In conclusion, the spikes in neuron σ_{in} are processed as follows. When there are some groups of $3T$ spikes of type a in neuron σ_{in}, rule $a^{2T}(a^{3T})^+/a^{3T} \to a^{3T}(1)$ or rule $a^{2T}(a^{3T})^+(b^{3T})^+/a^{3T} \to a^{3T}(1)$ or rule $a^{2T}b^T(a^{3T})^+(b^{3T})^+/a^{3T} \to a^{3T}(1)$ can be used first in order to store all the $3Tg(x)$ spikes of type a in neuron σ_1. Then all the $3Ty$ spikes of type b in neuron σ_{in} are passing into neuron $\sigma_{l_{i1}}$ by rule $a^{2T}(b^{3T})^+/b^{3T} \to b^{3T}(2)$ or rule $a^{2T}b^T(b^{3T})^+/b^{3T} \to b^{3T}(2)$. When all this is done, neuron σ_{in} finally sends $3T$ spikes of type a to neuron $\sigma_{l_{i1}}$. Besides, the spikes in neuron $\sigma_{l_{i1}}$ are processed as follows. $3Ty$ spikes of type a are sent to neuron σ_2 first by rule $(b^{3T})^+/b^{3T} \to a^{3T}(1)$ or rule $a^{3T}(b^{3T})^+/b^{3T} \to a^{3T}(1)$ in neuron $\sigma_{l_{i1}}$. When

there are only $3T$ spikes of type a left in neuron $\sigma_{l_{i1}}$, rule $a^{3T} \to b^{3T}(2)$ can be applied and the neuron emits $3T$ spikes of type b to neuron σ_{l_0}.

All in all, we can find that $3Tg(x)$ spikes of type a will be stored in neuron σ_1 and $3Ty$ spikes of type a will be stored in neuron σ_2 after neuron σ_{in} reads the spike train $a^{2T}(a^{3T})^{g(x)}(b^{3T})^y b^T$ from the environment. And finally, neuron σ_{l_0} will receive 3T spikes of type b. This means there is no "block" situation in this INPUT module.

A total of 67 neurons are used in this system: 8 neurons for 8 registers; 23 neurons for 23 instruction labels; 1 auxiliary neuron for each ADD module, 9 in total; 2 auxiliary neurons for each SUB module, 26 in total; INPUT module uses 1 auxiliary neuron.

From the description above, we know that ASNP-MCS systems with 67 neurons can correctly simulate register machine M_u working in computing mode. Therefore, Theorem 4.3 holds. □

5 CONCLUSIONS AND FUTURE WORK

In this work, we investigated a variant of SNP systems working in the asynchronous mode, asynchronous spiking neural P systems with multiple channels and symbols (ASNP-MCS systems for short). Different from regular SNP systems, ASNP-MCS systems work in the asynchronous mode: neurons are free to fire when their rules can be applied; if the coming of new spikes causes the rules to be unavailable, the computation will continue in the new configuration. Different from the existing asynchronous SNP systems, ASNP-MCS systems have two interesting characters: multiple channels and multiple symbols. We proved that ASNP-MCS systems as number generating and accepting devices are Turing universal. Then, we constructed an ASNP-MCS system with 67 neurons as a small universal function computing device.

Several open problems still need to be discussed. For example, how to avoid the "block" situation in asynchronous systems. In the existing work, Song et al. [41] provided a scheme for solving this problem with the mode of "synchronization". This paper has made some attempt to deal with this "block" problem. The features of multiple channels and multiple symbols can give some help, in accepting/computing mode, to solve the "block" situation in INPUT module. Some issues remain to investigate: Can the "block" be solved in SUB module? Is there a small universal ASNP-MCS system with less neurons?

From the perspective of application, ASNP-MCS systems are the distributed and parallel computation systems working in the asynchronous mode and have the feature of multiple channels and the power of dealing with multiple symbols. In this work, only the discussion of Turing universality of the systems was in our focus. In the future, we will consider the application of ASNP-MCS systems in complex problems, for example, high-order dynamic systems and social network in real-world.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (No. 62076206), and the Research Foundation of the Education Department of Sichuan Province (No. 17TD0034), China.

REFERENCES

[1] Păun, G.: Computing with Membranes. Journal of Computer and System Sciences, Vol. 61, 2000, No. 1, pp. 108–143, doi: 10.1006/jcss.1999.1693.

[2] Păun, G.—Rozenberg, G.—Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press, New York, 2010.

[3] Freund, R.—Păun, G.—Pérez-Jiménez, M. J.: Tissue P Systems with Channel States. Theoretical Computer Science, Vol. 330, 2005, No. 1, pp. 101–116, doi: 10.1016/j.tcs.2004.09.013.

[4] Bernardini, F.—Gheorghe, M.: Population P Systems. Journal of Universal Computer Science, Vol. 10, 2004, No. 5, pp. 509–539, doi: 10.3217/jucs-010-05-0509.

[5] Cienciala, L.—Ciencialová, L.: Some New Results of P Colonies with Bounded Parameters. Natural Computing, Vol. 17, 2018, No. 2, pp. 321–332, doi: 10.1007/s11047-016-9591-0.

[6] Ionescu, M.—Păun, G.—Yokomori, T.: Spiking Neural P Systems. Fundamenta Informaticae, Vol. 71, 2006, No. 2, pp. 279–308.

[7] Ciencialová, L.—Csuhaj-Varjú, E.—Kelemenová, A.—Vaszil, G.: Variants of P Colonies with Very Simple Cell Structure. International Journal of Computers Communications and Control, Vol. 4, 2009, No. 3, pp. 224–233, doi: 10.15837/ijccc.2009.3.2430.

[8] Păun, G.—Pérez-Jiménez, M. J.: Solving Problems in a Distributed Way in Membrane Computing: dP Systems. International Journal of Computers Communications and Control, Vol. 5, 2010, No. 2, pp. 238–250, doi: 10.15837/ijccc.2010.2.2478.

[9] Song, B.—Pan, L.: The Computational Power of Tissue-Like P Systems with Promoters. Theoretical Computer Science, Vol. 641, 2016, pp. 43–52, doi: 10.1016/j.tcs.2016.05.022.

[10] Valencia-Cabrera, L.—Orellana-Martín, D.—Martínez-Del-Amor, M. A.—Riscos-Núñez, A.—Pérez-Jiménez, M. J.: Computational Efficiency of Minimal Cooperation and Distribution in Polarizationless P Systems with Active Membranes. Fundamenta Informaticae, Vol. 153, 2017, No. 1-2, pp. 147–172, doi: 10.3233/fi-2017-1535.

[11] Zhang, X.—Liu, Y.—Luo, B.—Pan, L.: Computational Power of Tissue P Systems for Generating Control Languages. Information Sciences, Vol. 278, 2014, No. 10, pp. 285–297, doi: 10.1016/j.ins.2014.03.053.

[12] García-Quismondo, M.—Nisbet, I. C. T.—Mostello, C.—Reed, M. J.: Modeling Population Dynamics of Roseate Terns (Sterna dougallii) in the Northwest Atlantic Ocean. Ecological Modelling, Vol. 368, 2018, pp. 298–311, doi: 10.1016/j.ecolmodel.2017.12.007.

[13] Gheorghe, M.—Manca, V.—Romero-Campero, F. J.: Deterministic and Stochastic P Systems for Modelling Cellular Processes. Natural Computing, Vol. 9, 2010, No. 2, pp. 457–473, doi: 10.1007/s11047-009-9158-4.

[14] Zhang, G.—Rong, H.—Neri, F.—Pérez-Jiménez, M. J.: An Optimization Spiking Neural P System for Approximately Solving Combinatorial Optimization Problems. International Journal of Neural Systems, Vol. 24, 2014, No. 5, Art. No. 1440006, pp. 1–16, doi: 10.1142/s0129065714400061.

[15] Peng, H.—Shi, P.—Wang, J.—Riscos-Núñez, A.—Pérez-Jiménez, M. J.: Multiobjective Fuzzy Clustering Approach Based on Tissue-Like Membrane Systems. Knowledge-Based Systems, Vol. 125, 2017, pp. 74–82, doi: 10.1016/j.knosys.2017.03.024.

[16] Peng, H.—Wang, J.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: An Unsupervised Learning Algorithm for Membrane Computing. Information Sciences, Vol. 304, 2015, No. 20, pp. 80–91, doi: 10.1016/j.ins.2015.01.019.

[17] Peng, H.—Wang, J.—Shi, P.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: An Extended Membrane System with Active Membranes to Solve Automatic Fuzzy Clustering Problems. International Journal of Neural Systems, Vol. 26, 2016, No. 3, Art. No. 1650004, pp. 1–17, doi: 10.1142/s0129065716500040.

[18] Díaz-Pernil, D.—Berciano, A.—Peña-Cantillana, F.—Gutiérrez-Naranjo, M. A.: Segmenting Images with Gradient-Based Edge Detection Using Membrane Computing. Pattern Recognition Letters, Vol. 34, 2013, No. 8, pp. 846–855, doi: 10.1016/j.patrec.2012.10.014.

[19] Díaz-Pernil, D.—Gutiérrez-Naranjo, M. A.—Peng, H.: Membrane Computing and Image Processing: A Short Survey. Journal of Membrane Computing, Vol. 1, 2019, pp. 58–73, doi: 10.1007/s41965-018-00002-x.

[20] Wang, J.—Shi, P.—Peng, H.: Membrane Computing Model for IIR Filter Design. Information Sciences, Vol. 329, 2016, pp. 164–176, doi: 10.1016/j.ins.2015.09.011.

[21] Li, B.—Peng, H.—Wang, J.—Huang, X.: Multi-Focus Image Fusion Based on Dynamic Threshold Neural P Systems and Surfacelet Transform. Knowledge-Based Systems, Vol. 196, 2020, Art. No. 105794, pp. 1–12, doi: 10.1016/j.knosys.2020.105794.

[22] Li, B.—Peng, H.—Luo, X.—Wang, J.—Song, X.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: Medical Image Fusion Method Based on Coupled Neural P Systems in Nonsubsampled Shearlet Transform Domain. International Journal of Neural Systems, Vol. 31, 2021, No. 1, Art. No. 2050050, doi: 10.1142/S0129065720500501.

[23] Li, B.—Peng, H.—Wang, J.: A Novel Fusion Method Based on Dynamic Threshold Neural P Systems and Nonsubsampled Contourlet Transform for Multi-Modality Medical Images. Signal Processing, Vol. 178, 2021, Art. No. 107793, pp. 1–13, doi: 10.1016/j.sigpro.2020.107793.

[24] Pan, L.—Wang, J.—Hoogeboom, H. J.: Spiking Neural P Systems with Astrocytes. Neural Computation, Vol. 24, 2012, No. 3, pp. 805–825, doi: 10.1162/neco_a_00238.

[25] Păun, G.: Spiking Neural P Systems with Astrocyte-Like Control. Journal of Universal Computer Science, Vol. 13, 2007, No. 11, pp. 1707–1721.

[26] Pan, L.—Păun, G.: Spiking Neural P Systems with Anti-Spikes. International Journal of Computers Communications and Control, Vol. 4, 2009, No. 3, pp. 273–282, doi: 10.15837/ijccc.2009.3.2435.

[27] Wang, J.—Hoogeboom, H. J.—Pan, L.—Păun, G.—Pérez-Jiménez, M. J.: Spiking Neural P Systems with Weights. Neural Computation, Vol. 22, 2010, No. 10, pp. 2615–2646, doi: 10.1162/neco_a_00022.

[28] Zeng, X.—Zhang, X.—Song, T.—Pan, L.: Spiking Neural P Systems with Thresholds. Neural Computation, Vol. 26, 2014, No. 7, pp. 1340–1361, doi: 10.1162/neco_a_00605.

[29] Peng, H.—Chen, R.—Wang, J.—Song, X.—Wang, T.—Yang, F.—Sun, Z.: Competitive Spiking Neural P Systems with Rules on Synapses. IEEE Transactions on NanoBioscience, Vol. 16, 2017, No. 8, pp. 888–895, doi: 10.1109/tnb.2017.2783890.

[30] Song, T.—Pan, L.—Păun, G.: Spiking Neural P Systems with Rules on Synapses. Theoretical Computer Science, Vol. 529, 2014, pp. 82–95, doi: 10.1016/j.tcs.2014.01.001.

[31] Song, T.—Pan, L.: Spiking Neural P Systems with Rules on Synapses Working in Maximum Spiking Strategy. IEEE Transactions on NanoBioscience, Vol. 14, 2015, No. 4, pp. 465–477, doi: 10.1109/TNB.2015.2402311.

[32] Peng, H.—Yang, J.—Wang, J.—Wang, T.—Sun, Z.—Song, X.—Lou, X.—Huang, X.: Spiking Neural P Systems with Multiple Channels. Neural Networks, Vol. 95, 2017, pp. 66–71, doi: 10.1016/j.neunet.2017.08.003.

[33] Song, X.—Wang, J.—Peng, H.—Ning, G.—Sun, Z.—Wang, T.—Yang, F.: Spiking Neural P Systems with Multiple Channels and Anti-Spikes. Biosystems, Vol. 167–170, 2018, pp. 13–19, doi: 10.1016/j.biosystems.2018.05.004.

[34] Peng, H.—Wang, J.: Coupled Neural P Systems. IEEE Transactions on Neural Networks and Learning Systems, Vol. 30, 2019, No. 6, pp. 1672–1682, doi: 10.1109/TNNLS.2018.2872999.

[35] Peng, H.—Wang, J.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: Dynamic Threshold Neural P Systems. Knowledge-Based Systems, Vol. 163, 2019, pp. 875–884, doi: 10.1016/j.knosys.2018.10.016.

[36] Wu, T.—Păun, A.—Zhang, Z.— Pan, L.: Spiking Neural P Systems with Polarizations. IEEE Transactions on Neural Networks and Learning Systems, Vol. 29, 2018, No. 8, pp. 3349–3360, doi: 10.1109/TNNLS.2017.2726119.

[37] Peng, H.—Li, B.—Wang, J.—Song, X.—Wang, T.—Valencia-Cabrera, L.—Pérez-Hurtado, I.—Riscos-Núñez, A.—Pérez-Jiménez, M. J.: Spiking Neural P Systems with Inhibitory Rules. Knowledge-Based Systems, Vol. 188, 2020, Art. No. 105064, pp. 1–10, doi: 10.1016/j.knosys.2019.105064.

[38] Peng, H.—Bao, T.—Luo, X.—Wang, J.—Song, X.—Riscos-Núñez, A.—Pérez-Jiménez, M. J.: Dendrite P Systems. Neural Networks, Vol. 127, 2020, pp. 110–120, doi: 10.1016/j.neunet.2020.04.014.

[39] Peng, H.—Lv, Z.—Li, B.—Luo, X.—Wang, J.—Song, X.—Wang, T.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: Nonlinear Spiking Neural P Systems. International Journal of Neural Systems, Vol. 30, 2010, No. 10, Art. No. 2050008, pp. 1–17, doi: 10.1142/S0129065720500082.

[40] Cavaliere, M.—Ibarra, O. H.—Păun, G.—Egecioglu, O.—Ionescu, M.—Woodworth, S.: Asynchronous Spiking Neural P Systems. Theoretical Computer Science, Vol. 410, 2009, No. 24-25, pp. 2352–2364, doi: 10.1016/j.tcs.2009.02.031.

[41] Song, T.—Pan, L.—Păun, G.: Asynchronous Spiking Neural P Systems with Local Synchronization. Information Sciences, Vol. 219, 2013, No. 10, pp. 197–207, doi: 10.1016/j.ins.2012.07.023.

[42] Song, T.—Pan, L.—Jiang, K.—Song, B.—Chen, W.: Normal Forms for Some Classes of Sequential Spiking Neural P Systems. IEEE Transactions on NanoBioscience, Vol. 12, 2013, No. 3, pp. 255–264, doi: 10.1109/tnb.2013.2271278.

[43] Pan, L.—Păun, G.—Zhang, G.—Neri, F.: Spiking Neural P Systems with Communication on Request. International Journal of Neural Systems, Vol. 27, 2017, No. 8, Art. No. 1750042, pp. 1–13, doi: 10.1142/s0129065717500423.

[44] Zhang, X.—Pan, L.—Păun, A.: On The Universality of Axon P Systems. IEEE Transactions on Neural Networks and Learning Systems, Vol. 26, 2015, No. 11, pp. 2816–2829, doi: 10.1109/tnnls.2015.2396940.

[45] Peng, H.—Wang, J.—Pérez-Jiménez, M. J.—Wang, H.—Shao, J.—Wang, T.: Fuzzy Reasoning Spiking Neural P System for Fault Diagnosis. Information Sciences, Vol. 235, 2013, No. 20, pp. 106–116, doi: 10.1016/j.ins.2012.07.015.

[46] Wang, J.—Shi, P.—Peng, H.—Pérez-Jiménez, M. J.—Wang, T.: Weighted Fuzzy Spiking Neural P Systems. IEEE Transactions on Fuzzy Systems, Vol. 21, 2013, No. 2, pp. 209–220, doi: 10.1109/tfuzz.2012.2208974.

[47] Peng, H.—Wang, J.—Ming, J.—Shi, P.—Pérez-Jiménez, M. J.—Yu, W.—Tao, C.: Fault Diagnosis of Power Systems Using Intuitionistic Fuzzy Spiking Neural P Systems. IEEE Transactions on Smart Grid, Vol. 9, 2018, No. 5, pp. 4777–4784, doi: 10.1109/tsg.2017.2670602.

[48] Peng, H.—Wang, J.—Shi, P.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: Fault Diagnosis of Power Systems Using Fuzzy Tissue-Like P Systems. Integrated Computer-Aided Engineering, Vol. 24, 2017, No. 4, pp. 401–411, doi: 10.3233/ica-170552.

[49] Wang, T.—Zhang, G.—Zhao, J.—He, Z.—Wang, J.—Pérez-Jiménez, M. J.: Fault Diagnosis of Electric Power Systems Based on Fuzzy Reasoning Spiking Neural P Systems. IEEE Transactions on Power Systems, Vol. 30, 2015, No. 3, pp. 1182–1194, doi: 10.1109/TPWRS.2014.2347699.

[50] Wang, J.—Peng, H.—Yu, W.—Ming, J.—Pérez-Jiménez, M. J.—Tao, C.—Huang, X.: Interval-Valued Fuzzy Spiking Neural P Systems for Fault Diagnosis of Power Transmission Networks. Engineering Applications of Artificial Intelligence, Vol. 82, 2019, pp. 102–109, doi: 10.1016/j.engappai.2019.03.014.

[51] Macías, L. F.—Pérez-Hurtado, I.—García-Quismondo, M.—Valencia-Cabrera, L.—Pérez-Jiménez, M. J.—Riscos-Núñez, A.: A P-Lingua Based Simulator for Spiking Neural P Systems. In: Gheorghe, M., Păun, G., Rozenberg, G., Salomaa, A., Verlan, S. (Eds.): Membrane Computing (CMC 2011). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7184, 2012, pp. 257–281, doi: 10.1007/978-3-642-28024-5_18.

[52] Cabarle, F. G. C.—Adorna, H.—Martínez, M. A.: A Spiking Neural P System Simulator Based on CUDA. In: Gheorghe, M., Păun, G., Rozenberg, G., Salomaa, A., Verlan, S. (Eds.): Membrane Computing (CMC 2011). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7184, 2012, pp. 87–103, doi: 10.1007/978-3-642-28024-5_8.

[53] Song, T.—Zou, Q.—Liu, X.—Zeng, X.: Asynchronous Spiking Neural P Systems with Rules on Synapses. Neurocomputing, Vol. 151, 2015, Part 3, pp. 1439–1445, doi: 10.1016/j.neucom.2014.10.044.

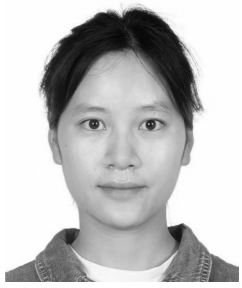
[54] Cabarle, F. G. C.—Adorna, H. N.—Pérez-Jiménez, M. J.—Song, T.: Spiking Neural P Systems with Structural Plasticity. Neural Computing and Applications, Vol. 26, 2015, No. 8, pp. 1905–1917, doi: 10.1007/s00521-015-1857-4.

[55] Song, T.—Liu, X.—Zeng, X.: Asynchronous Spiking Neural P Systems with Anti-Spikes. Neural Processing Letters, Vol. 42, 2015, No. 3, pp. 633–647, doi: 10.1007/s11063-014-9378-1.

[56] Cavaliere, M.—Egecioglu, O.—Ibarra, O. H.—Ionescu, M.—Păun, G.—Woodworth, S.: Asynchronous Spiking Neural P Systems: Decidability and Undecidability. In: Garzon, M. H., Yan, H. (Eds.): DNA Computing (DNA 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4848, 2007, pp. 246–255, doi: 10.1007/978-3-540-77962-9_26.

[57] Zhang, X.—Zeng, X.—Pan, L.: On Languages Generated by Asynchronous Spiking Neural P Systems. Theoretical Computer Science, Vol. 410, 2009, No. 26, pp. 2478–2488, doi: 10.1016/j.tcs.2008.12.055.

[58] Korec, I.: Small Universal Register Machines. Theoretical Computer Science, Vol. 168, 1996, No. 2, pp. 267–301, doi: 10.1016/s0304-3975(96)00080-1.



Wenmei Yi received her B.Sc. degree in computer science from Sichuan Police College, Luzhou, China in 2016. She is currently pursuing her M.Sc. degree with School of Computer and Software Engineering, Xihua University, Chengdu, China. Her current research interests include membrane computing and machine learning.

Zeqiong Lv received her B.Sc. degree in computer science from Xihua University, Chengdu, China, in 2018. She is currently pursuing her M.Sc. degree with School of Computer and Software Engineering, Xihua University, Chengdu, China. Her current research interests include membrane computing and machine learning.

Hong Peng received his Ph.D. degree in signal and information processing from the University of Electronic Science and Technology of China, Chengdu, China in 2010. He has been Professor with School of Computer and Software Engineering since 2005. His research interests include membrane computing, machine learning, image processing.

Xiaoxiao Song received his Ph.D. degree in electrical engineering from the Chongqing University, Chongqing, China in 2010. He is currently Associate Professor with School of Electrical and Information Engineering, Xihua University, China. His research interests include membrane computing and image processing.

Jun Wang received her Ph.D. degree in electrical engineering from the Southwest Jiaotong University, Chengdu, China in 2006. She was Lecturer with the Sichuan College of Science and Technology, China, from 1991 to 2003. She was Associate Professor with the Xihua University, China, from 1998 to 2003. Since 2004 she has been Professor with the School of Electrical and Information Engineering. Her research interests include electrical automation, intelligent control, and membrane computing.

Computing and Informatics, Vol. 39, 2020, 952–972, doi: 10.31577/cai_2020_5_952

A MAPREDUCE ALGORITHM FOR MINIMUM VERTEX COVER PROBLEMS AND ITS RANDOMIZATION

Morikazu Nakamura, Daiki Kinjo, Takeo Yoshida

Faculty of Engineering
University of the Ryukyus
Okinawa 903-0213, Japan
e-mail: morikazu@ie.u-ryukyu.ac.jp

Abstract. MapReduce is a programming paradigm for large-scale distributed information processing. This paper proposes a MapReduce algorithm for the minimum vertex cover problem, which is known to be NP-hard. The MapReduce algorithm can efficiently obtain a minimal vertex cover in a small number of rounds. We show the effectiveness of the algorithm through experimental evaluation and comparison with exact and approximate algorithms which demonstrates a high quality in a small number of MapReduce rounds. We also confirm from experimentation that the algorithm has good scalability allowing high-quality solutions under restricted computation times due to increased graph size. Moreover, we extend our algorithm to randomized one to obtain a good expected approximate ratio.

Keywords: MapReduce, minimum vertex cover, Hadoop, optimization, algorithm design, randomized algorithm

1 INTRODUCTION

MapReduce is a programming paradigm introduced by Google [1] as a promising software platform for large-scale distributed information processing. MapReduce uses functional programming and is composed of *mappers* for per-record computation and *reducers* for results aggregation [2]. MapReduce platforms can be implemented on a large number of commodity computers or computer clusters which provides scalability and fault tolerance – the most important characteristics required

for large-scale distributed processing [3, 4]. The number of processing nodes can be easily increased to handle growing large data.

Spark is another distributed computing platform for big-data analysis [5]. Compared to Hadoop, a well-known MapReduce platform, Spark's in-memory computation is high-speed [6, 7]. On the other hand, Hadoop was designed for efficiency in cost and time. This mechanism, based on an enormous volume of data on several storage nodes leads to outstanding scalability with lower costs, even though the real-time computation is sacrificed. Although Hadoop and Spark are often compared, their roles are differentiated, and they can coexist mutually [6, 7].

Large graphs are often used for modeling various real life objects, systems, and services, for example, road networks, relations among SNS (social network service) members, citation networks of research papers, the hyperlink structure of web pages, and various relations among pieces of digital content on the Internet. The number of vertices in such graphs may be several million, hundreds of millions, several billion, or even more. For such huge graphs, structured data mining requires graph algorithms such as breadth-first search (BFS), depth-first search (DFS), minimum spanning tree (MST), or shortest path problem (SPP). Traditional computing platforms and paradigms are not suited to this task because they are insufficiently scalable. Therefore, MapReduce algorithms for graph problems are an important research field [8, 9, 10, 11, 12, 13, 14, 15]. Many application areas of this topic are expected in engineering, biology, and the medical sciences [16, 17]. Several MapReduce programming platforms have been so far developed [13, 14, 15, 18, 19, 20] that provide APIs for graph operations and show how to implement some basic algorithms, such as page ranking, SPP, and MST, by using those APIs. There are also MapReduce algorithms for the maximum clique problem [21], the maximum cover problem [22], the maximum flow problem [23], and the shortest-path problem [24].

The minimum vertex cover problem (MVC) is a basic problem in graph theory, and it is well known to be computationally intractable, that is, NP-hard [25]. Approximate solutions with a two-approximation ratio can be easily constructed from maximal matching, and then the approximation factor has been slightly improved [26, 27].

We propose a MapReduce algorithm for MVC, a greedy algorithm that drastically improves the solution quality compared to maximal matching-based algorithms. In our previous works, we presented the first version of the algorithm without detailed experimental evaluation and deep discussion [28]. Moreover, in this paper we extend the original algorithm into randomized one to avoid the worst case solution quality happened in specific situations.

In [32], MapReduce algorithms for well-known problems are proposed, where they show the theoretical approximation ratio is two for the minimum vertex cover problem. On the other hand, our first algorithm may lead to solutions of worse quality for special cases than the MapReduce algorithm in the literature. However our randomized version can overcome such worst case situation and obtain solutions with expected approximation ratio 4/3. Moreover we show by experimental eval-

uations that obtained solutions are quite better than the expected approximation ratio.

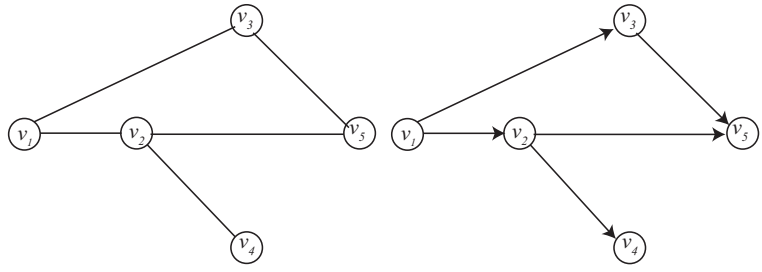
The remainder of the paper is organized as follows. In Section 2, we briefly give some basic background on graphs, MVC and MapReduce. In Section 3, we propose our MapReduce algorithm for MVC and show its correctness. In Section 4, we perform a computational experiment to evaluate our proposal. In Section 5, a randomized MapReduce algorithm is presented. Finally, in Section 6, we conclude the paper with some remarks.

2 PRELIMINARIES

This section explains some graph notations, the definition of a minimum vertex cover, and a basic background on MapReduce for the readability of the remaining paper.

2.1 Graphs

Graphs are denoted by a 2-tuple $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ with vertex set $\mathbf{V}$ and edge set $\mathbf{E}$, where the elements of $\mathbf{E}$ are 2-element subsets of $\mathbf{V}$. If each edge in $\mathbf{E}$ has an orientation (that is, a direction), the graph becomes a directed graph and is shown as $\vec{\mathbf{G}} = (\mathbf{V}, \vec{\mathbf{E}})$. Figure 1 a) shows the graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ where $\mathbf{V}=\{v_1, v_2, v_3, v_4, v_5\}$ and $\mathbf{E} = \{(v_1, v_2), (v_1, v_3), (v_2, v_4), (v_2, v_5), (v_3, v_5)\}$. Figure 1 b) shows a directed graph whose underlying graph is the graph in (a).



a) Undirected graph b) Directed graph

Figure 1. Graph

The degree $\mathsf{d}(v)$ of a vertex v is the number of edges connected to v. In the case of directed graphs, the in-degree $\mathsf{d}^{in}(v)$ and the out-degree $\mathsf{d}^{out}(v)$ of a vertex v are defined as the number of incoming and outgoing edges, respectively. Examples are $\mathsf{d}(v_1) = 2$, $\mathsf{d}(v_2) = 3$ in Figure 1 a) and $\mathsf{d}^{in}(v_2) = 1$ and $\mathsf{d}^{out}(v_2) = 2$ in Figure 1 b). A matching $\mathbf{M}$ of a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is a subset of $\mathbf{E}$ such that no pair of edges in $\mathbf{M}$ shares a vertex. A matching is maximal if adding any edge not in $\mathbf{M}$ results in no longer being a matching. A vertex cover $\mathbf{VC}$ of a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is a set of

vertices such that each edge in **E** is incident to at least one vertex in **VC**. For example, in Figure 1, $\{(v_1, v_3), (v_2, v_5)\}$ is a maximal matching and $\{v_2, v_3\}$ is a vertex cover. More detailed information on graphs can be referred to the literature [31, 30].

2.2 Minimum Vertex Cover Problem

The minimum vertex cover problem is a classical graph problem and is known to be NP-hard. The problem is formulated for an input graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ with the vertex set $\mathbf{V} = \{1, 2, \ldots, n\}$ and the edge set $\mathbf{E} = \{(v, u) | v \text{ and } u \in \mathbf{V}\}$ as an integer programming problem.

MVC (Integer Programming):

$$\min \sum_{v \in \mathbf{V}} x_v \tag{1}$$

s.t.

$$x_u + x_v \geq 1, \forall (u, v) \in \mathbf{E}, \tag{2}$$

$$x_v \in \{0, 1\}. \tag{3}$$

For the problem, a factor-2 approximate solution can be obtained from a maximal matching constructed by a simple greedy algorithm. Figure 2 gives an approximate solution. The dotted edges express a maximal matching. We can easily confirm that all endpoints of the dotted edges, the vertices with bold lines, comprise a vertex cover of the graph. The approximation factor is no more than two since the minimum vertex cover should include at least one endpoint of each edge in the matching.

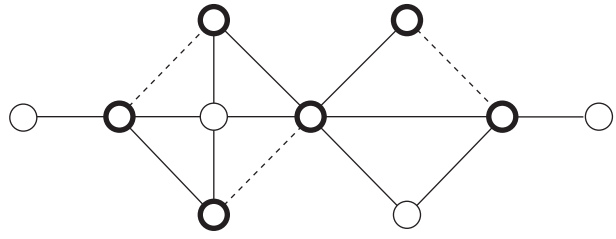


Figure 2. Approximation solution based on maximal matching

2.3 MapReduce Model

MapReduce is a framework for large-scale distributed computing based on the well-known master-slave parallel processing pattern [1]. MapReduce programs are composed of map operations and reduce operations. Map operations play a role in the

per-record computation and Reduce operations in the results aggregation. In Map operations, the master divides the input data into multiple data sets and assigns these sets to worker nodes. Note that the input data can be stored beforehand in distributed data storage near worker nodes to reduce the overhead incurred by data division and assignment. Each worker processes the assigned data according to the map operation and returns the output to its master node. In Reduce operations, the master lets workers collect the outputs of the map operations and combine them to form the answer to the problem. Workers perform these operations in parallel, taking key–value pairs as processing primitives. The MapReduce processing can be formally explained as follows.

For sets of the input key–value pairs $U_0^i, i = 0, 1, \ldots, m-1$, the MapReduce program performs the following steps.

For $t = 1, 2, 3, \ldots, T$ do

1. **Execute Map**: Input each pair (k, v) in U_{t-1}^i to mapper M_i, $i = 0, 1, \ldots, m-1$ and the mapper performs the Map operation. Each mapper generates a sequence of pairs, (k_1, v_1), (k_2, v_2), $\ldots$, as the result. Let us denote the multiset of key–value pairs generated by M_i at t^{th} round by $\hat{U}_t^i$, used in **Shuffle** step.
2. **Shuffle**: For each k, let $V_{k,i}$ be the multiset of values v_j such that $(k, v_j) \in \hat{U}_t^i$. The MapReduce system constructs the multiset $V_{k,i}$ from $\hat{U}_t^i$.
3. **Execute Reduce**: For each k, input k and some arbitrary permutation of $V_{k,i}$ to reducer $R_i, i = 0, 1, \ldots, r-1$ and perform the Reduce operation. The reducer generates a sequence of key–value pairs (k, v_1'), (k, v_2'), $\ldots$, as the result. Let U_t^i, $i = 0, 1, \ldots, m-1$ be the multiset of key–value pairs generated by R_j, $j = 0, 1, \ldots, r-1$.

MapReduce is a promising platform for distributed large-scale computation. However, algorithms for MapReduce are quite different from ordinary algorithms we have used. Therefore, we need to design suitable algorithms for the platform.

3 MAPREDUCE ALGORITHM FOR MVC

In this section we propose a MapReduce algorithm for the minimum vertex cover problem. Algorithm 1 shows a pseudocode for the algorithm, MapReduceMVC.

Before going to the explanation, we introduce some notations. Functions $\mathsf{N} : \mathbf{V} \to 2^{|\mathbf{V}|}$ and $\mathsf{d} : \mathbf{V} \to \{0, \ldots, |\mathbf{V}| - 1\}$ return the set of the vertices neighboring v and degree of $v, v \in \mathbf{V}$, respectively. Function $\mathsf{index} : \mathbf{V} \to \{1, \ldots, |\mathbf{V}|\}$ returns the index of v.

A relation $\prec$ on $\mathbf{V}$ and directed graphs induced by $\prec$ is defined as follows:

Definition 1. Let $\mathbf{G}$ be an undirected graph with vertex set $\mathbf{V}$ and edge set $\mathbf{E}$. Let us denote by $v \prec u$ the relation such that $(\mathsf{d}(v), \mathsf{index}(v))$ is smaller than $(\mathsf{d}(u), \mathsf{index}(u))$ in the lexicographical order on $\{0, \ldots, |V| - 1\} \times \{1, \ldots, |V|\}$.

Algorithm 1 Pseudocode for MapReduce algorithm

procedure MapReduceMVC:
−−STEP1:
$VC[v] \leftarrow$ **true**, for all v;
$Adj[v] \leftarrow$ **true**, for all v;
−−STEP2:
if v is source **then**
  $VC[v] \leftarrow$ **false**;
end if
repeat
  −−STEP3:
  if $VC[u] =$ **true**, for all $u \in \mathsf{N}(v)$ **then**
    $Adj[v] \leftarrow$ **true**;
  else
    $Adj[v] \leftarrow$ **false**;
  end if
  −−STEP4:
  if $VC[v] =$ **true and** $Adj[v] =$ **true and** $\exists u \in \mathsf{N}(v),\ Adj[u] =$ **false then**
    $VC[v] \leftarrow$ **false**;
  end if
  if $VC[v] =$ **false and** $Adj[v] =$ **false then**
    Let N^f be $\{u | u \in \mathsf{N}(v), VC(u) =$ **false**$\}$;
    if $u \prec v, \exists u \in \mathsf{N}^f$ **then**
      $VC[v] \leftarrow$ **true**;
    end if
  end if
until No modification is taken place in STEP4
−−STEP5:
$x_v \leftarrow 1$ if $VC[v] =$ **true**, otherwise $x_v \leftarrow 0$, for all $v \in \mathbf{V}$;

Definition 2. For an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, its directed graph induced by $\prec$, $\vec{\mathbf{G}} = (\mathbf{V}, \vec{\mathbf{E}})$, is constructed by orienting each edge $(v, u) \in \mathbf{E}$ from v to u when $v \prec u$.

The following lemma is straightforwardly obtained from the definition since the relation $\prec$ is transitive.

Lemma 1. Let $\vec{\mathbf{G}} = (\mathbf{V}, \vec{\mathbf{E}})$ be the directed graph induced by $\prec$ for a given undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Then $\vec{\mathbf{G}}$ is an acyclic graph.

Definition 3. For an acyclic directed graph $\vec{\mathbf{G}} = (\mathbf{V}, \vec{\mathbf{E}})$, $v \in \mathbf{V}$ is called a *source* if $\mathsf{d}^{in}(v) = 0$.

Figure 3 shows an example of the edge orientation, where the number in each vertex v represents $\mathsf{index}(v)$. Figure 3 b) shows the directed graph constructed from

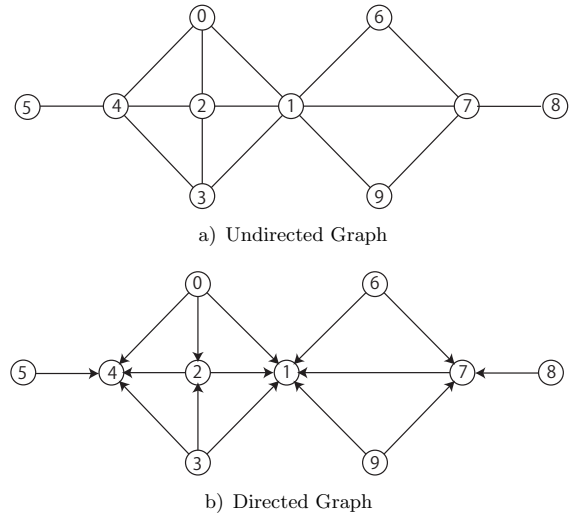


Figure 3. Example of edge orientation

the undirected graph in Figure 3 a). The vertices 0, 3, 5, 6, 8, and 9 are sources of the acyclic graph. In MapReduceMVC, each worker node can compute the relation $\prec$ between the assigned vertex and its neighbors since workers know the index and the degree of all the neighbors.

Arrays of *boolean* variables $VC[v]$ and $Adj[v]$ indicate whether or not v is in the vertex cover and whether or not all the neighbor vertices of v are in the cover, respectively. At the end of the algorithm (Line 28), a vertex cover is constructed by collecting all vertices v such that $VC[v] = \textbf{true}$.

The algorithm is composed with five STEPs and each STEP corresponds to one MapReduce operation. No worker can proceed to the next STEP (MapReduce operation) before all the workers complete the current operation.

At STEP1 (Lines 3 and 4), $VC[v]$ and $Adj[v]$ are initialized to **true** for all the vertices. Note that our algorithm initially adds all the vertices into the vertex cover, then gradually excludes unnecessary vertices in STEP3 and 4.

STEP2 (Lines 6 to 8) sets $VC[v]$ to **false** for each v if v is a source. Each source vertex becomes a trigger to reduce the size of the vertex cover from the initial vertex cover constructed at STEP1.

STEP3 (Lines 11 to 15) and STEP4 (Lines 17 to 25) are iteratively executed in the repeat-until statement. STEP3 updates $Adj[v]$ for all vertices. $Adj[v]$ needs to be updated whenever there is a neighbor $u \in \mathsf{N}(v)$ such that $VC[u]$ was changed in the previous iteration.

STEP4 is composed of two parts. The first part (Lines 17 to 19) is for improving the vertex cover by attempting to decrease its size. The second part (Lines 20 to 25) checks and maintains feasibility of the solution. No solution is feasible when there

exists a pair v and u such that $(v, u) \in \mathbf{E}$, yet $VC[v] = VC[u] =$ **false**. In such case, $VC[v]$ is set to **true** if $u \prec v$, otherwise $VC[u]$ is set to **true** to repair its infeasibility.

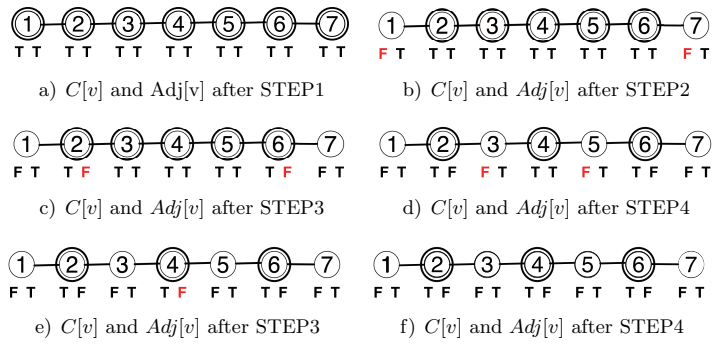


Figure 4. Demonstration of MapReduceMVC for a line graph

Figure 4 shows a demonstration example where we depict the steps of the algorithm when applied to a line graph with seven vertices. In the figure, a pair of letters such as T T, T F, or F T (where T means **true** and F means **false**) below a vertex corresponds to a pair of values of $VC[v]$ and $Adj[v]$. Time proceeds from top to bottom in the figure. The pair of *boolean* values at each vertex is initialized to T T by STEP1 shown in Figure 4 a). Both vertices 1 and 7 are sources in this example, and so become F T at STEP2. Figure 4 c) shows that $Adj[v]$ becomes **false** for each of vertices 2 and 6 since 1 and 7 are removed from the vertex cover at Figure 4 b). In Figure 4 d), each of vertices 3 and 5 is removed from the vertex cover since $Adj[v]$ for 2 and 6 was changed to **false**. In Figure 4 e), $Adj[v]$ for vertex 4 is updated. Finally in (f), we confirmed no modification is performed where each vertex should be either T F or F T. All vertices with T F are included in the vertex cover obtained by the algorithm.

We show now the validity of our algorithm by the following lemma and theorem.

Lemma 2. MapReduceMVC completes its execution after the k^{th} STEP4 if and only if $VC[v] =$ **true** and $Adj[v] =$ **false** or $VC[v] =$ **false** and $Adj[v] =$ **true**, for all $v \in \mathbf{V}$ at the end of the k^{th} STEP3, where k is a natural number.

Proof.

Sufficiency: We assume the following condition holds: For all $v \in \mathbf{V}$, $VC[v] =$ **true** and $Adj[v] =$ **false** or $VC[v] =$ **false** and $Adj[v] =$ **true** after the k^{th} STEP3. Considering synchronous execution of MapReduce operations, $Adj[v]$, for all $v \in \mathbf{V}$, has a correct value after STEP3, that is, $Adj[v] =$ **true** when $VC[u] =$ **true**, for all $u \in \mathsf{N}[v]$, otherwise $Adj[v] =$ **false**. At the kth STEP4, $VC[v]$ can be modified when $VC[v] =$ **true** and $Adj[v] =$ **true** or $VC[v] =$ **false**

and $Adj[v] =$ **false** but not otherwise. Therefore, no modification occurs at the k^{th} STEP4, so the algorithm can break the repeat condition and complete its execution.

Necessity: We assume the algorithm stops its execution and there exists a node v such that $VC[v] =$ **true** and $Adj[v] =$ **true** or $VC[v] =$ **false** and $Adj[v] =$ **false**.

(CASE1: Suppose that $VC[v] = Adj[v] =$ **true**, $\exists v \in \mathbf{V}$.) for all $u \in \mathsf{N}[v]$, $VC[u] =$ **true** since $Adj[v] =$ **true** and also for all $u \in \mathsf{N}[v], Adj[u] =$ **true** because the condition at STEP4 (Line 17) does not hold at the end of the algorithm. By repeating the same consideration, we finally find that for all $v \in \mathbf{V}, VC[v] = Adj[v] =$ **true**. However, according to Lemma 1, at STEP2 there exists at least one source vertex v that lets $VC[v] =$ **false**. Only Line 20 in STEP4 can change $VC[v]$ from **false** to **true**, and it is performed only when there exists a neighbor u such that $VC[v] = VC[u] =$ **false**. Moreover, only one side of both vertices v and u can be changed from **false** to **true** according to the order relation $\prec$. Therefore there should exist at least one vertex v such that $VC[v] =$ **false** even if such situations occur successively on adjacent vertices. This situation contradicts the first assumption.

(CASE2: Suppose that $VC[v] = Adj[v] =$ **false**, $\exists v \in \mathbf{V}$.) There exists at least one vertex $u \in \mathsf{N}(v)$ such that $VC[u] =$ **false** since $Adj[v] =$ **false**. Moreover, $Adj[u] =$ **false** because $VC[v] =$ **false**. This situation holds the condition of the second *if* statement in STEP4. Therefore, the algorithm cannot break the *repeat-until loop* statement. This situation contradicts the first assumption. □

Theorem 1. MapReduceMVC outputs a minimal vertex cover for a given graph.

Proof. From Lemma 2, for all $v \in \mathbf{V}$, $VC[v] =$ **true** and $Adj[v] =$ **false** or $VC[v] =$ **false** and $Adj[v] =$ **true** after the algorithm stops its execution. Therefore, the output of the algorithm should generate a vertex cover since $VC[u] =$ **true** for all $u \in \mathsf{N}(v)$ when $VC[v] =$ **false**. It is also obvious that no proper subset of the vertex cover obtained by the algorithm can be a vertex cover, that is, it shows minimality. □

4 EXPERIMENTAL EVALUATION

We implemented our MapReduce algorithm under Hadoop [3] and evaluated its solution quality and the number of MapReduce rounds. For comparison purposes, we developed a Hadoop program of the maximal matching-based vertex cover algorithm described above.

Test graph data were randomly generated based on two topological characteristics: random graphs and scale-free graphs. A graph is scale-free if its degree distribution follows a power law. Scale-free graphs are known as a model often observed in actual networks [29]. Random graphs with average degree d were generated such that vertex degrees follow the Guassian distribution with average d and variance 1.

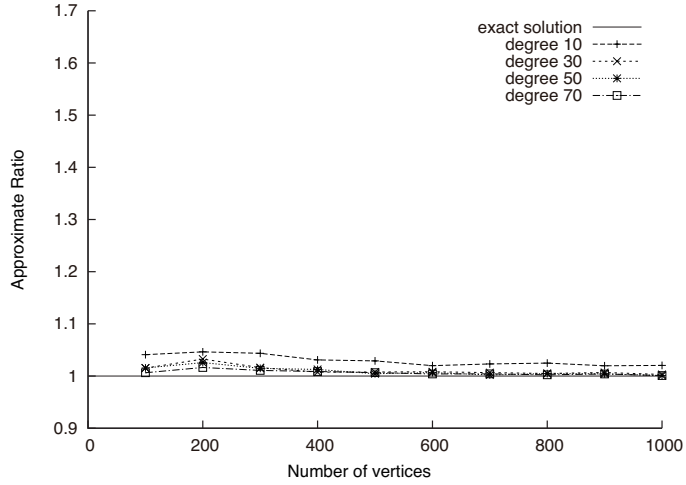


Figure 5. Solution quality vs. graph size for random graphs (our proposal)

4.1 Solution Quality

We first evaluate the solution quality of our MapReduce algorithm by comparing it with its maximal matching-based algorithm. The Gurobi optimizer, a commercial solver, was used to obtain exact solutions but it could not exactly solve problem instances of huge size random graphs within a reasonable time. We therefore eval-

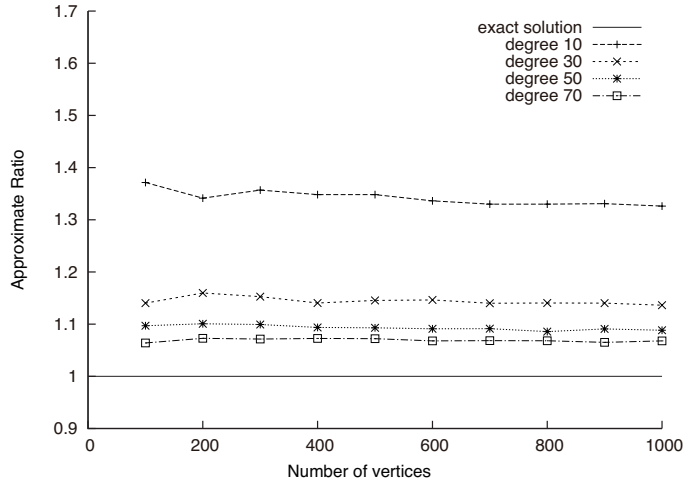


Figure 6. Solution quality vs. graph size for random graphs (maximal matching-based algorithm)

uated, for random graphs of size 100 to 1 000, the quality of approximate solutions by comparing with that of exact solutions, while for random graphs with 1 000 to 10 000 vertices, we just compared both approximate algorithms, our proposal and the maximal matching-based algorithm.

When it comes to scale-free graphs, the Gurobi optimizer was able to solve problem instances of 1 000 000 vertices. This is because the exact algorithm can drastically prune branches of the search tree for the power-law distribution of edge degree in the scale-free graphs. Therefore, for scale-free graphs, we varied the number of vertices up to 1 000 000 to compare with exact solutions.

Figures 5 and 6, respectively, depict the solution quality of our algorithm and that of the matching-based algorithm.

In the figures, the horizontal axis shows the number of vertices and the vertical axis shows the approximate ratio compared to the exact solution. Here we take the quality of the exact solution as 1. For the experiments, we varied the average degree of a vertex from 10 to 70 and prepared ten different instances of each. Therefore, the values in the figures show the average of ten executions. Figures 7 and 8 show results for larger random graphs, where the curves express the size of the obtained minimal vertex covers for each degree. These figures indicate that both algorithms obtained lower quality solutions for smaller degrees of random graphs. That is, loosely-coupled random graphs are harder to solve approximately than are tightly-coupled ones. The results indicate that our algorithm can obtain quite good solutions of less than 5 % approximate ratio, even for degree 10, while the approximate ratio of the matching-based algorithm was more than 30 % in case of the degree 10.

For scale-free graphs, we compare both approximate algorithms with the exact algorithm and depict the quality versus the graph size in Figure 9. The results confirm that our algorithm performs very well also for scale-free graphs compared to the matching-based algorithm.

Through experimental evaluation, we confirmed the effectiveness of our proposed algorithm. From a quality point of view, our algorithm outperformed the maximal matching-based algorithm. The maximal matching-based algorithm progressively constructs a maximal matching, then generates a minimal vertex cover from the maximal matching. The algorithm is based on a greedy policy for the maximality of matching, but not for the minimality of the vertex cover. In contrast, our algorithm focuses on minimizing the size of the vertex cover. Our algorithm generates an initial vertex cover that includes only $|V| - |\text{Sources}|$ vertices, and then removes unnecessary vertices step-by-step in a greedy manner. This direct greedy operation greatly improves the solution quality as compared with the maximal matching-based algorithm.

4.2 Number of Rounds

Instead of measuring real computation time of MapReduce operations, we usually evaluate the number of rounds of MapReduce operations. Actual computation time is much more understandable for performance evaluations, but it strongly depends

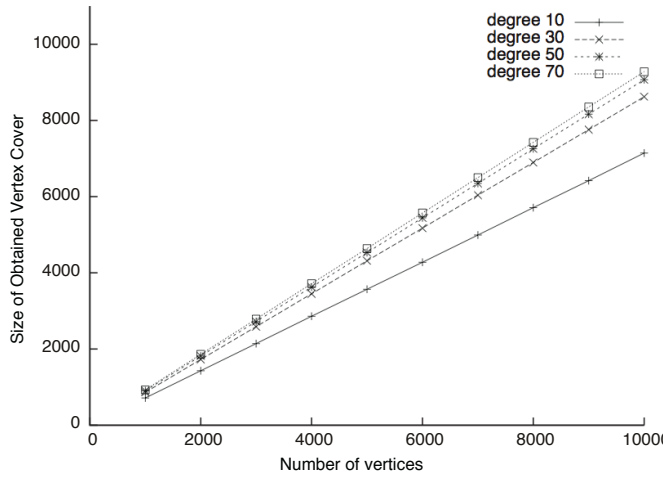


Figure 7. Results for large random graphs (our proposal)

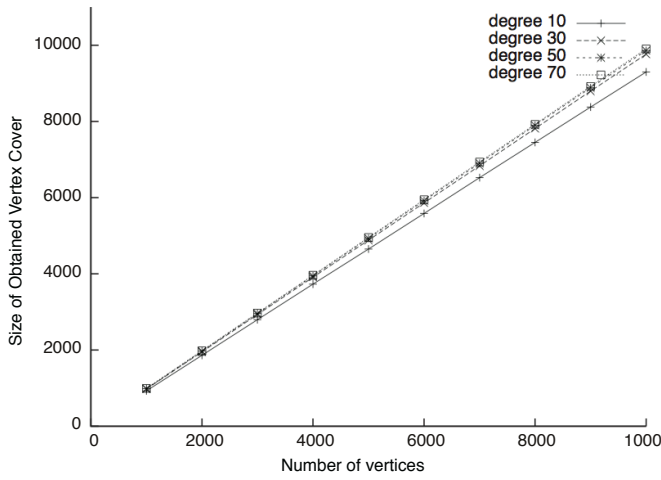


Figure 8. Results for large random graphs (maximal matching-based algorithm)

on conditions of the computing platform, such as the number of processing nodes, cpu spec, memory size, network architecture, and traffic situation. Therefore, the number of rounds becomes the most reliable factor by which to evaluate the computation time of MapReduce programs. In our experiment, we measured the number of rounds of two approximate algorithms. Figures 10 and 11 respectively depict the number of rounds used in our algorithm and the maximal matching-based algorithm for random graphs. Figure 12 compares the number of rounds for scale-free graphs.

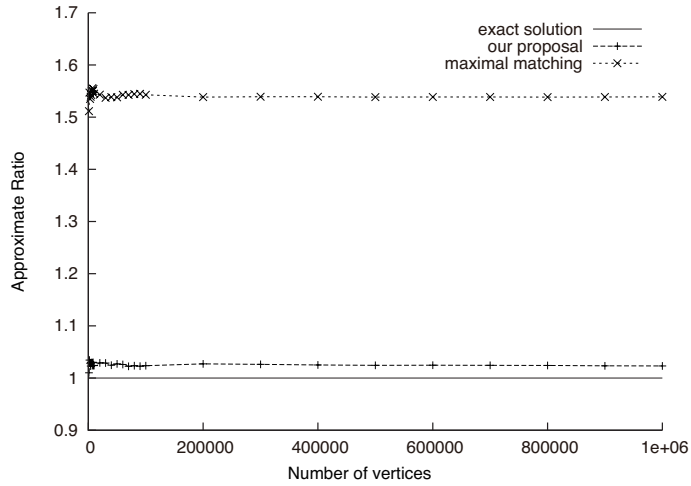


Figure 9. Solution quality vs. graph size for scale-free graphs (our proposal)

The results indicate that our algorithm requires relatively few rounds, compared to the maximal matching-based algorithm. The real computation time is proportional to the number of rounds in the experiment and the curves of the number of rounds are almost constant with respect to the number of vertices. Therefore, our algorithm has good scalability regarding graph size.

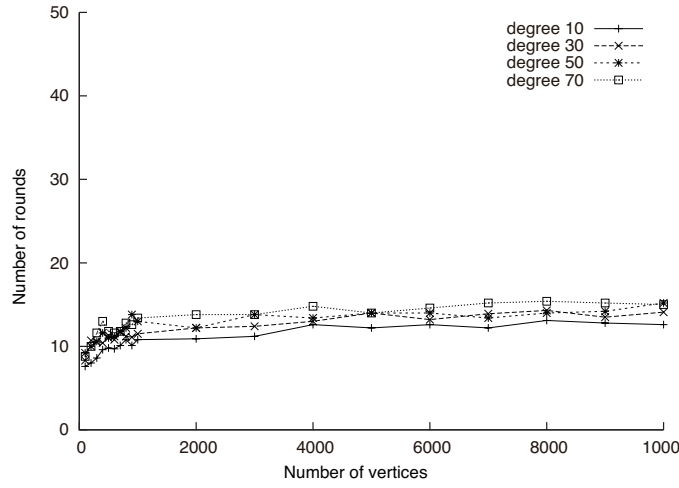


Figure 10. Number of rounds vs. graph size for random graphs (our proposal)

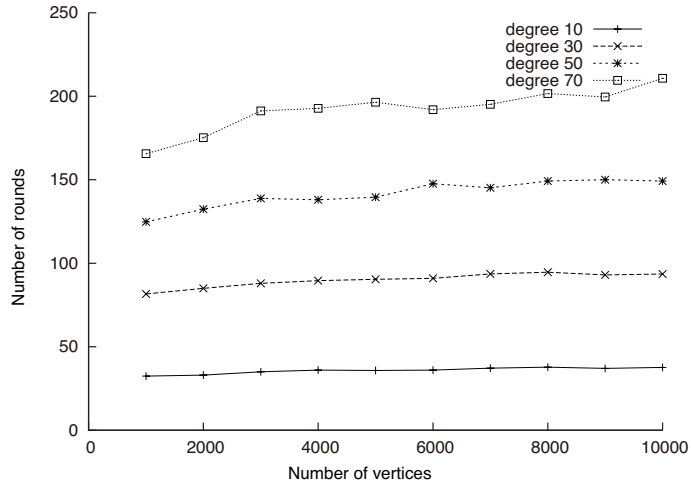


Figure 11. Number of rounds vs. graph size for random graphs (maximal matching-based algorithm)

5 RANDOMIZED MAPREDUCEMVC

MapReduceMVC is not always highly efficient. We consider here a special class of graphs, called *k-flower graphs* for which our original MapReduceMVC may not output good quality of solutions.

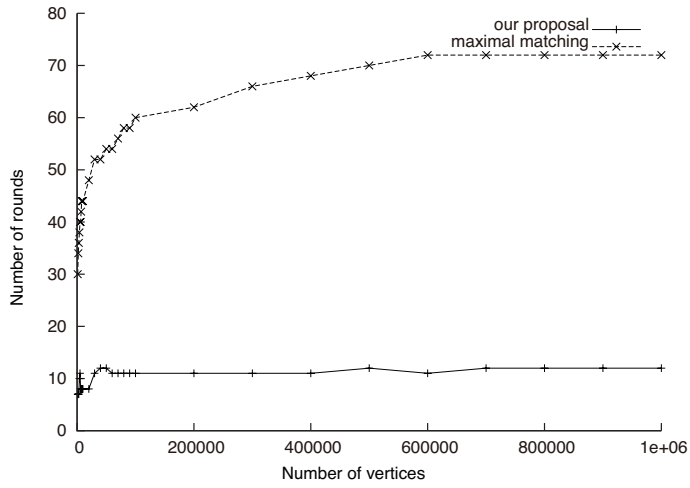


Figure 12. Number of rounds vs. graph size for scale-free graphs

Definition 4. A graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ is called a k-flower graph when $\mathbf{V} = \{v_0, v_1, v_2, \ldots, v_{3\cdot(k-1)+3}\}$ and $\mathbf{E} = \cup_{i=1}^{k}\{(v_0, v_{3\cdot(i-1)+1}), (v_{3\cdot(i-1)+1}, v_{3\cdot(i-1)+2}), (v_{3\cdot(i-1)+2}, v_{3\cdot(i-1)+3}), (v_{3\cdot(i-1)+3}, v_0)\}$.

Definition 5. For k-flower graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, the vertex v_0 is called *center vertex* and subgraph $\mathbf{G}(i) = (\mathbf{V}(i), \mathbf{E}(i)), i = 1, 2, \ldots, k$, is called a petal of k-flower graph $\mathbf{G}$ where $\mathbf{V}(i) = \{v_0, v_{3\cdot(i-1)+1}, v_{3\cdot(i-1)+2}, v_{3\cdot(i-1)+3}\}$, $\mathbf{E}(i) = \{(v_0, v_{3\cdot(i-1)+1}), (v_{3\cdot(i-1)+1}, v_{3\cdot(i-1)+2}), (v_{3\cdot(i-1)+2}, v_{3\cdot(i-1)+3}), (v_{3\cdot(i-1)+3}, v_0)\}$.

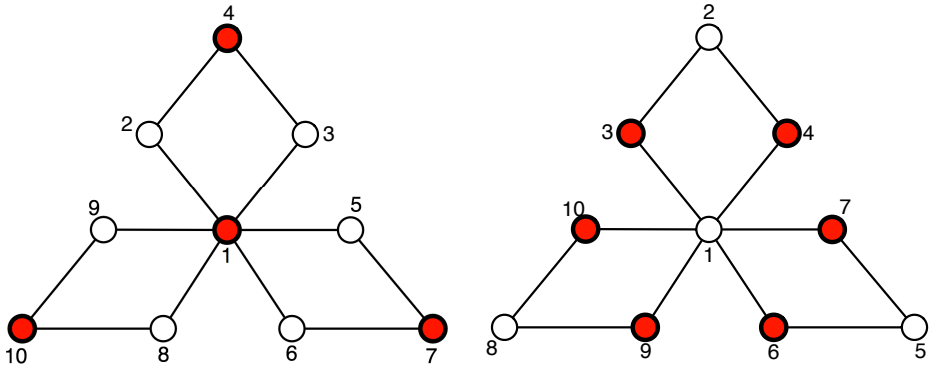
Figure 13 depicts 3-flower graph and its exact and the worst case vertex covers, where the vertices colored in red denote the vertices in the obtained vertex cover and the number beside a vertex shows its index. Figures 13 a) and 13 b) represent the exact solution and the worst case solution obtained by our MapReduce algorithm, respectively. From the definition, we can easily prove that the size of the optimal solution and the worst case solution for k-flower graphs are

$$|\mathbf{VC}^{\mathsf{best}}| = k + 1, \tag{4}$$

$$|\mathbf{VC}^{\mathsf{worst}}| = 2 \cdot k + 1, \tag{5}$$

respectively. Therefore, the approximation ratio for k-flower graphs is

$$\frac{|\mathbf{VC}^{\mathsf{worst}}|}{|\mathbf{VC}^{\mathsf{best}}|} = \frac{2 \cdot k + 1}{k + 1} \approx 2. \tag{6}$$



a) Optimal solution b) Worst case solution

Figure 13. Solution example for 3-flower graphs

Randomized algorithms allow us to avoid the worst case solution, where we utilize only randomized index function in our MapReduceMVC. Our algorithm requires the construction of a directed graph for the input graph by the edge orientation

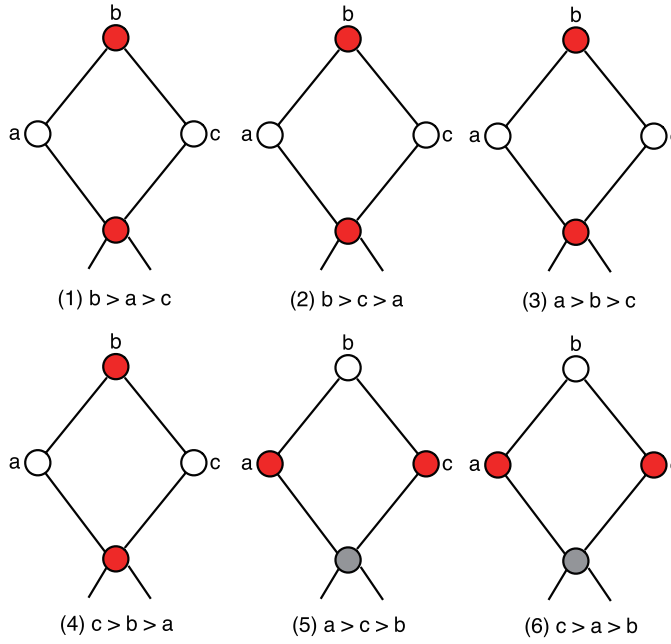


Figure 14. All patterns of randomized index

based on the lexicographical order on $(\prec, \mathsf{index})$. In k-flower graphs, the edge orientation depends on the index values of the terminal vertices since all the vertices except for the center vertex have degree 2. This is the reason why it is hard for our original MapReduceMVC to solve k-flower graphs.

Let us calculate the expected size of minimal vertex covers to be obtained by the randomized MapReduceMVC for k-flower graphs. Figure 14 shows all the patterns of indexing for one petal of a k-flower graph, where a, b, and c express the index value for the corresponding vertex. The vertex with the smallest index number out of a, b, and c should be a source which can not be in the final vertex cover. The vertices colored in red are in the vertex cover. From the figures, the expected value of the number of vertices in $\mathbf{VC}$ per petal except for the vertex v_0, $E(|\mathbf{VC}(petal)|$ is

$$E(|\mathbf{VC}(petal)|) = 1 \cdot \frac{2}{3} + 2 \cdot \frac{1}{3} = \frac{4}{3}. \tag{7}$$

Since v_0 is in $\mathbf{VC}$ only when the indexing is either $a > c > b$ or $c > a > b$ for all the petals, the expected value for the center vertex in $\mathbf{VC}$, $E(|\mathbf{VC}(v_0)|$ is

$$E(|\mathbf{VC}(v_0)|) = 1 - \left(\frac{1}{3}\right)^k. \tag{8}$$

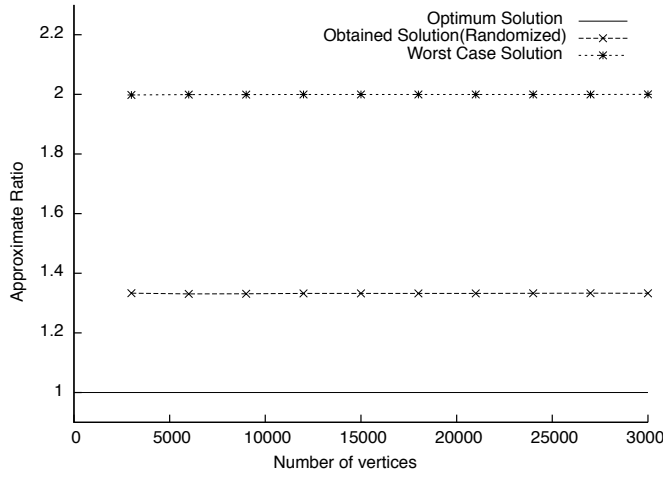


Figure 15. Solution quality of randomized MapReduceMVC for k-flower graphs

Therefore, the expected value of $|\mathbf{VC}|$ is

$$E(|\mathbf{VC}|) = k \cdot E(|VC(petal)|) \tag{9}$$

$$= \frac{4}{3} \cdot k + 1 - \left(\frac{1}{3}\right)^k. \tag{10}$$

The expected approximation ratio is as follows:

$$E\left(\frac{|\mathbf{VC}|}{|\mathbf{VC}^{\mathsf{best}}|}\right) = \frac{\frac{4}{3} \cdot k + 1 - \left(\frac{1}{3}\right)^k}{k+1} \approx \frac{4}{3}. \tag{11}$$

To confirm the validity of the expected approximate ratio, we performed an experiment. Figure 15 shows the average curves of the approximate ratio in which we solved MVC for k-flower graphs 20 times by the Randomized MapReduceMVC with varying the number of vertices. From the figures, our randomized MapReduceMVC could achieve the same approximate ratio as the one we calculated in (11). Our randomized algorithm can be easily implemented since we just need to introduce a randomized index function to the original one.

Theorem 2. RandomizedMapReduceMVC outputs a minimal vertex cover with the expected approximate ratio 4/3 for k-flower graphs.

6 CONCLUSION

We proposed a MapReduce algorithm for the minimum vertex cover problem and proved its validity. We performed an experimental evaluation of our proposal and

measured the quality of solutions and the number of rounds of MapReduce operations. We observed that our algorithm could generate a reasonable quality of approximate solutions compared to the exact algorithm for random graphs with 100 to 1 000 vertices and scale-free graphs with 1 000 to 1 000 000 vertices. We also compared our algorithm with the well-known maximal matching-based minimum vertex cover algorithm, and our algorithm outperformed it not only in terms of solution quality but also in terms of computation time.

Finally we introduced a class of graphs, k-flower graphs, which it is hard for our algorithm to solve, and we have proposed a randomized version of MapReduceMVC. Just by using the randomized index function, our algorithm can obtain the expected approximate ratio 4/3 for k-flower graphs.

REFERENCES

[1] Dean, J.—Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI '04), 2004, pp. 137–150.

[2] Karloff, H.—Suri, S.—Vassilvitskii, S.: A Model of Computation for MapReduce. Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, 2010, pp. 938–948, doi: 10.1137/1.9781611973075.76.

[3] Holmes, A.: Hadoop in Practice. Manning Publications Co., Greenwich, 2012.

[4] Guhther, N. J.—Puglia, P.—Tomasette, K.: Hadoop Superlinear Scalability. Communications of the ACM, Vol. 58, 2015, No. 4, pp. 46–55, doi: 10.1145/2719919.

[5] Apache Spark. `https://spark.apache.org/`.

[6] Ketu, S.—Mishra, P. K.—Agarwal, S.: Performance Analysis of Distributed Computing Frameworks for Big Data Analytics: Hadoop vs. Spark. Computación y Sistemas, Vol. 24, 2020, No. 2, pp. 669–689, doi: 10.13053/cys-24-2-3401.

[7] Mostafaeipour, A.—Jahangard Rafsanjani, A.—Ahmadi, M.—Arockia Dhanraj, J.: Investigating the Performance of Hadoop and Spark Platforms on Machine Learning Algorithms. The Journal of Supercomputing, Vol. 77, 2021, pp. 1273–1300, doi: 10.1007/s11227-020-03328-5.

[8] Lin, J.—Schatz, M.: Design Patterns for Efficient Graph Algorithms in MapReduce. Proceedings of the Eighth Workshop on Mining and Learning with Graphs. 2010, pp. 78–85, doi: 10.1145/1830252.1830263.

[9] Warashina, T.—Aoyama, K.—Sawada, H.—Hattori, T.: Efficient K-Nearest Neighbor Graph Construction Using MapReduce for Large-Scale Data Sets. IEICE Transactions on Information and Systems, Vol. E97.D, 2014, No. 12, pp. 3142–3154, doi: 10.1587/transinf.2014edp7108.

[10] Devi, N. S.—Mane, A. C.—Mishra, S.: Computational Complexity of Minimum P4 Vertex Cover Problem for Regular and K1, 4-Free Graphs. Discrete Applied Mathematics, Vol. 184, 2015, pp. 114–121, doi: 10.1016/j.dam.2014.10.033.

[11] Guruswami, V.—Sachdeva, S.—Saket, R.: Inapproximability of Minimum Vertex Cover on k-Uniform k-Partite Hypergraphs. SIAM Journal on Discrete Mathematics, Vol. 29, 2013, No. 1, pp. 36–58, doi: 10.1137/130919416.

[12] Lattanzi, S.—Moseley, B.—Suri, S.—Vassilvitskii, S.: Filtering: A Method for Solving Graph Problems in MapReduce. Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '11), 2011, pp. 85–94, doi: 10.1145/1989493.1989505.

[13] Plimpton, S. J.—Devine, K. D.: MapReduce in MPI for Large-Scale Graph Algorithms. Parallel Computing, Vol. 37, 2011, No. 9, pp. 610–632, doi: 10.1016/j.parco.2011.02.004.

[14] Malewicz, G.—Austern, M. H.—Bik, A. J. C.—Dehnert, J. C.—Horn, I.—Leiser, N.—Czajkowski, G.: Pregel: A System for Large-Scale Graph Processing. Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10), 2010, pp. 135–146, doi: 10.1145/1807167.1807184.

[15] Kambatla, K.—Rapolu, N.—Jagannathan, S.—Grama, A.: Asynchronous Algorithms in MapReduce. Proceedings of 2010 IEEE International Conference of Cluster Computing, Heraklion, Greece, 2010, pp. 245–254, doi: 10.1109/cluster.2010.30.

[16] Bellettini, C.—Camilli, M.—Capra, L.—Monga, M.: State Space Exploration of RT Systems in the Cloud. Cornell University Library, arXiv:1203.6806 [cs.SE], 2012, 6 pp.

[17] Taylor, R. C.: An Overview of the Hadoop/MapReduce/HBase Framework and Its Current Applications in Bioinformatics. BMC Bioinformatics, Vol. 11, 2010, No. S1, doi: 10.1186/1471-2105-11-s12-s1.

[18] Liang, F.—Lu, X.: Accelerating Iterative Big Data Computing Through MPI. Journal of Computer Science and Technology, Vol. 30, 2015, No. 2, pp. 283–294, doi: 10.1007/s11390-015-1522-5.

[19] Yu, W. K.—Wang, Y. D.—Que, X. Y.—Xu, C.: Virtual Shuffling for Efficient Data Movement in MapReduce. IEEE Transactions on Computers, Vol. 64, 2015, No. 2, pp. 556–568, doi: 10.1109/tc.2013.216.

[20] Kang, U.—Tsourakakis, C. E.—Faloutsos, C.: PEGASUS: Mining Peta-Scale Graphs. Knowledge and Information Systems, Vol. 27, 2011, No. 2, pp. 303–325, doi: 10.1007/s10115-010-0305-0.

[21] Wu, B.—Yang, S.—Zhao, H.—Wang, B.: A Distributed Algorithm to Enumerate All Maximal Cliques in MapReduce. Proceedings of the 2009 Fourth International Conference on Frontier of Computer Science and Technology, 2009, pp. 45–51, doi: 10.1109/fcst.2009.30.

[22] Chierichetti, F.—Kumar, R.—Tomkins, A.: Max-Cover in Map-Reduce. Proceedings of the 19th International Conference on World Wide Web (WWW '10), 2010, pp. 231–240, doi: 10.1145/1772690.1772715.

[23] Halim, F.—Yap, R. H. C.—Wu, Y.: A MapReduce-Based Maximum-Flow Algorithm for Large Small-World Network Graphs. Proceedings of the 2011 31st International Conference on Distributed Computing Systems, 2011, pp. 192–202, doi: 10.1109/icdcs.2011.62.

[24] Kumar, P.—Singh, A. K.: MapReduce Algorithm for Single Source Shortest Path Problem. International Journal of Computer Network and Information Security (IJCNIS), Vol. 12, 2020, No. 3, pp. 11–21, doi: 10.5815/ijcnis.2020.03.02.

[25] Karp, R. M.: Reducibility Among Combinatorial Problems. In: Miller, R. E., Thatcher, J. W., Bohlinger, J. D. (Eds.): Complexity of Computer Computations. Springer, Boston, MA, The IBM Research Symposia Series, 1972, pp. 85–103, doi: 10.1007/978-1-4684-2001-2_9.

[26] Karakostas, G.: A Better Approximation Ratio for the Vertex Cover Problem. In: Caires, L., Italiano, G. F., Monteiro, L., Palamidessi, C., Yung, M. (Eds.): Automata, Languages and Programming (ICALP 2005). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3580, 2005, pp. 1043–1050, doi: 10.1007/11523468_84.

[27] Bar-Yehuda, R.—Even, S.: A Local-Ratio Theorem for Approximating the Weighted Vertex Cover Problem. In: Ausiello, G., Lucertini, M. (Eds.): Annals of Discrete Mathematics 25. North-Holland Mathematics Studies, Vol. 109, 1985, pp. 27–45, doi: 10.1016/s0304-0208(08)73101-3.

[28] Kinjo, D.—Nakamura, M.: A MapReduce Algorithm for Minimum Vertex Cover Problem. Proceedings of International Technical Conference on Circuits and Systems, Computers and Communications, 2013, pp. 505–508.

[29] Li, L.—Doyle, J. C.—Willinger, W.—Alderson, D.: Towards a Theory of Scale-Free Graphs: Definition, Properties, and Implications. Internet Mathematics, Vol. 2, 2005, No. 4, pp. 431–523, doi: 10.1080/15427951.2005.10129111.

[30] Bondy, J. A.—Murty, U. S. R.: Graph Theory with Applications. Elsevier Science Publishing, 1976.

[31] Diestel, R.: Graph Theory. Second Edition. Springer-Verlag, New York, 2000.

[32] Harvey, N. J. A.—Liaw, C.—Liu, P.: Greedy and Local Ratio Algorithms in the MapReduce Model. Proceedings of the 30$^{\text{th}}$ ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '18), 2018, pp. 43–52, doi: 10.1145/3210377.3210386.

Morikazu Nakamura received his B.E. and M.E. degrees from the University of the Ryukyus in 1989 and 1991, respectively, and D.E. degree from Osaka University in 1996. He is currently Professor in the area of computer science and intelligent systems, Faculty of Engineering, University of the Ryukyus, Japan. His research interests include theory and applications on mathematical systems. He is a member of IEICE and IEEE.

Daiki Kinjo received his B.E. and M.E. degrees in information engineering from the University of the Ryujyus in 2012 and 2014, respectively. He is currently Engineer in KLab Inc. His research interests include distributed computing, network computing, and data analysis.

Takeo Yoshida received his B.E. and M.E. degrees in electrical engineering from the Nagaoka University of Technology and the D.E. degree in electrical engineering from the Tokyo Metropolitan University in 1991, 1993 and 1997, respectively. He is currently Assistant Professor in the Department of Engineering, University of the Ryukyus, Japan. His research interests include dependable computing, VLSI design, and graph theory. He is a member of IEEE and IPSJ.

Computing and Informatics, Vol. 39, 2020, 973–993, doi: 10.31577/cai_2020_5_973

DEEP LSTM WITH GUIDED FILTER FOR HYPERSPECTRAL IMAGE CLASSIFICATION

Yanhui Guo, Fuli Qu*, Zhenmei Yu*, Qian Yu

School of Data and Computer Science
Shandong Women's University
2399 Daxue Road, Changqing District
Jinan, China
e-mail: {guoyanhui03, qufuli23}@163.com, {zhenmei_yu, yuqian}@sdwu.edu.cn

Abstract. Hyperspectral image (HSI) classification has been a hot topic in the remote sensing community. A large number of methods have been proposed for HSI classification. However, most of them are based on the extraction of spectral feature, which leads to information loss. Moreover, they rarely consider the correlation among the spectrums. In this paper, we see spectral information as a sequential data which should be relevant to each other. We introduce long short-term memory (LSTM) model, which is a typical recurrent neural network (RNN), to deal with HSI classification. To tackle the problem of overfitting caused by limited labeled samples, regularization strategy is introduced. For unbalance in different classes, we improve LSTM by weighted cost function. Also, we employ guided filter to smooth the HSI that can greatly improve the classification accuracy. And we proposed a method for modeling hyperspectral sequential data, which is very useful for future research work. Finally, the experimental results show that our proposed method can improve the classification performance as compared to other methods in three popular hyperspectral datasets.

Keywords: Recurrent neural network, long short-term memory, guided filter, hyperspectral image classification

* Corresponding authors

1 INTRODUCTION

Remote sensors can acquire hyperspectral images, which has hundreds of spectral data channels of the same pixel. The detailed spectral information can increase the recognition of materials, so as to improve the classification accuracy of materials. HSI classification has generated considerable attention and has been widely used in various areas including land cover, environmental protection, and agriculture. However, there are still two key issues that need to be addressed, curse of dimensionality and limited number of labeled training samples.

It is generally known that the task of HSI classification is to categorize the pixels into one of several classes by their spectral features. A large number of pixel-wise classifiers have been proposed to deal with the HSI classification, including random forests [7], k-Nearest Neighbor [19], support vector machine (SVM) [20], and sparse representation [4]. However, most of these traditional methods suffer from the Hughes phenomenon [18]. To solve the aforementioned problem, feature extraction and feature selection are adopted in these methods. Generally, principal component analysis (PCA) [8] and independent component analysis (ICA) [22] are common methods in feature extraction. Band selection [9] or subspace projection techniques [1] are widely adopted in classification of spectral patterns. Although these methods have improved classification accuracy, both feature extraction and subspace projection can lead to information loss and cannot make full use of hyperspectral features.

In recent years, deep learning has made promising progresses in many fields. Deep learning based methods also are adopted in HSI classification, including the autoencoder [3, 16], convolutional neural network (CNN) [2] and deep belief network (DBN) [5]. The paper [3] proposed a deep learning framework for HSI classification. Autoencoder learns to reconstruct the input vector and reduce the dimension of spectrum. Then a multi-class logistic regression was used to classify the HSI. CNN uses extensive parameter-sharing to tackle the curse of dimensionality and also classifies hyperspectral data directly in the spectral domain. Hu et al. [17] proposed a five-layers network, which can achieve better classification performance. Chen et al. [2] proposed a regularized 3-D CNN-based feature extraction model to extract efficient spectral-spatial features for HSI classification. Additionally, spectral-spatial classification was proposed by many researchers which combines spatial context with spectral information. However, this research is beyond the scope of this paper.

Autoencoder and CNN model obtain better classification accuracy in hyperspectral image, owing to their feature representation. Yet, there is a large number of parameters to be trained in the CNN model. For a hyperspectral image with only a small number of labeled samples, the advantages of CNN model cannot be fully realized. Moreover, all the aforementioned methods view the spectrum as a vector, and can result in information loss. The spectrum of a pixel is regarded as independent of each other. A pixel is considered a point in an orderless feature space. However, hyperspectral data is seen as a continuing spectra sequences in continuous spectrum bands. Recurrent neural network is a typical deep learning model for

solving sequential problems. RNN parameters are less, which is more suitable for the case of fewer training sets than CNN. So, we make use of a recurrent neural network (RNN) to characterize the sequential property for classifying the HSI.

Unlike object recognition, it is difficult to train a RNN model for reaching a steady state. We introduce guided filter to smooth the HSI, which greatly improves the HSI classification accuracy. Other filters (bilateral filter, joint bilateral filter) also can be used to smooth the noise. We adopt guided filter due to the ability to preserve edge information and gradient. The detailed steps are as follows:

1. We adopted a recurrent neural network (LSTM) for HSI classification. It is a very novel idea to regard spectral information as an ordered series. We learn the correlation between spectrums and LSTM, which is useful for HSI classification.
2. In order to improve the classification accuracy, we employ the guided filter to smooth HSI. Guided filter can denoise the image and preserve the edge of the image. So, it can greatly improve the classification accuracy.
3. Since the labeled samples are limited, deep learning model tends to overfit. To solve it, we adopt some regularization strategies, such as L2 regularization and dropout.
4. To address the problem of unbalance of the samples in different classes, we implemented a weighted cost function, which can improve the average accuracy of the classification.
5. The proposed methods are applied on two widely used hyperspectral datasets. We do compared experiments with SVM classifier and Autoencoder for three evaluation metrics.

The rest of this paper is organized as follows. Section 2 describes the related methodology and work. Section 3 gives the analysis of HSI classification and the proposed model in detail. Section 4 shows the results of the experiment, which is followed by conclusions in Section 5.

2 RELATED METHODOLOGY AND WORK

This section presents the principle of LSTM and guided filter, which is the foundation of this paper.

2.1 Recurrent Neural Networks

Recurrent neural networks (RNN) are a family of neural networks for processing sequential data, and have been successfully applied in many fields, such as natural language processing [21], speech recognition [12], image recognition [13], etc. RNN consists of an input layer, a hidden layer, and an output layer. Unlike traditional neural networks, there are links between hidden layer units. Given an input sequence

$x = (x_1, x_2, \ldots, x_T)$, a standard RNN computes the hidden vector sequence $h = (h_1, h_2, \ldots, h_T)$ and output vector sequence $y = (y_1, y_2, \ldots, y_T)$ by iterating the following equations from $t = 1$ to T:

$$h_t = g(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \tag{1}$$

$$y_t = W_{hy}h_t + b_y \tag{2}$$

where the W denotes weight matrices (e.g., W_{xh} is input-hidden weight matrix), the b denotes bias vectors (e.g., b_y is output bias vector) and g is the hidden layer function. Generally, g is a bounded function such as a logistic sigmoid function or a hyperbolic tangent function.

While RNN is focused on sequential relationship, and has made promising progresses in many fields, it still encounters difficulty in dealing with long-term sequential data since the gradients tend to vanish. An improved RNN model, named long short-term memory (LSTM) [11], is proposed for the above problem. The key to LSTM is the cell state, which takes the former state and current data as input. LSTM consists of five parts, including input gate, output gate, forget gate, cell input and cell output. The calculation process is as follows.

First, we compute the values for the input gate i_t, and the candidate value $\tilde{C}_t$ for the states of the memory cells at time t:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{3}$$

$$\tilde{C}_t = \tanh(W_{ic}x_t + U_c h_{t-1} + b_c). \tag{4}$$

Then, we compute the forget gate activation f_t at time t:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f). \tag{5}$$

Given the value of the input gate activation i_t, the forget gate activation f_t and the candidate state value $\tilde{C}_t$, we can compute the memory cells' new state C_t:

$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1}. \tag{6}$$

Finally, we can compute the value of their output gates and their outputs:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o), \tag{7}$$

$$h_t = o_t * \tanh(C_t) \tag{8}$$

where x_t is the input to the memory cell layer at time t, W, U and V denote the weight matrices, b is bias vector, σ is the activation function.

2.2 Guided Filter

Guided filter[1] was proposed for the first time by He [15]. Guided filter is widely used in noise reduction, image abstraction, etc. Given a guidance I and an input image p, we can obtain an output image q by guided filter. Generally, q is a linear transform of I in a window ω_k centered at the pixel k. If the radius of k is r, the size of local window ω_k is $(2r+1) \times (2r+1)$.

$$q_i = a_k I_i + b_k, \forall i \in \omega_k \tag{9}$$

where a_k is linear coefficient and b_k is a bias. From the model, it is obvious that $\nabla q = a\partial I$, which means that the filtering output q will have similar edge with guidance image I.

To obtain the coefficient and bias, a cost function for minimizing the term of mean error in the window ω_k is applied as follows:

$$E(a_k, b_k) = \Sigma_{i \in \omega_k}((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2). \tag{10}$$

Here, ϵ is a regularization parameter which could affect the blurring for the guided filter. According to the literal [10], formula (10) leads to a solution as follows.

$$a_k = \frac{\frac{1}{|\omega|}\sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon}, \tag{11}$$

$$b_k = \bar{p}_k - a_k \mu_k \tag{12}$$

where μ_k and σ_k^2 are the mean and variance of I in ω_k, $|\omega|$ is the number of pixels in the local window, and $\bar{p}_k$ is the mean of p in the window. After obtaining the coefficient a_k, b_k, we can compute the filtering output q_i. Through the above process, we can get a linear transform image q.

3 THE ANALYSIS AND MODELING OF HSI CLASSIFICATION

3.1 The Analysis of HSI Spectrums

LSTM model is better at dealing with sequence data. We analyze the feature of HSI from two dimensions. One is whether spectral information has sequence characteristics. The other is whether the spectral information of different classes is separable. We selected three categories in the KSC dataset (Figure 1). We can see that the hyperspectral data have sequence characteristics and have different spectral characteristics between each other to classify. This is the assumption based on which we examine the proposed idea in this paper.

[1] `http://kaiminghe.com/eccv10/index.html`

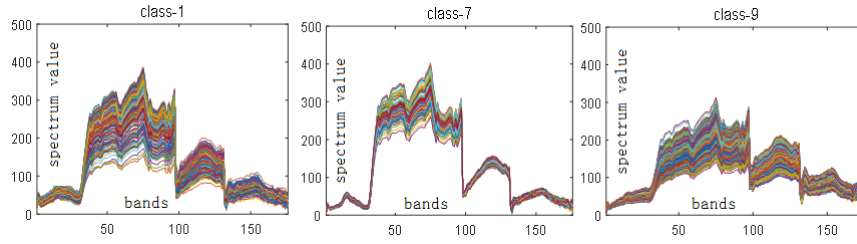


Figure 1. Spectral data of the 3 classes selected from KSC dataset with 176 channels

3.2 The Proposed Method of HSI Classification

We proposed a novel LSTM model for HSI classification with guided filter. First, we obtain a color guidance image from the original HSI by PCA method. Then, we filter the original HSI by guidance image. Finally, filtered HSI was classified by the improved LSTM model. The process is shown in Figure 2.

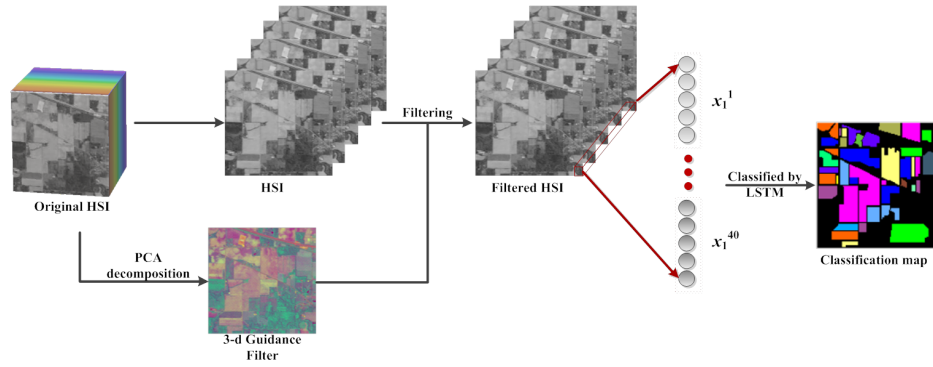


Figure 2. Framework of HSI classification by the proposed method

3.2.1 Filtering the HSI with Guided Filter

From Figure 1 (especially class-1), we can see that the regularity with spectral data is obvious. That is, materials in the same class have similar waveforms. However, there is still a lot of noise, inconsistent with the overall trend. Just like class-7, the right of the waveform is of some confusion. To solve this problem, we introduce guided filter to smooth the noises. Guided filter is an edge-preserving filter, which can be used for edge-aware smoothing. In this paper, the spectral data in Figure 1 were converted into the data in Figure 3 by guided filter. As Figure 3 shows, the spectral data is more obvious after filtering. The implementation of the method is as follows.

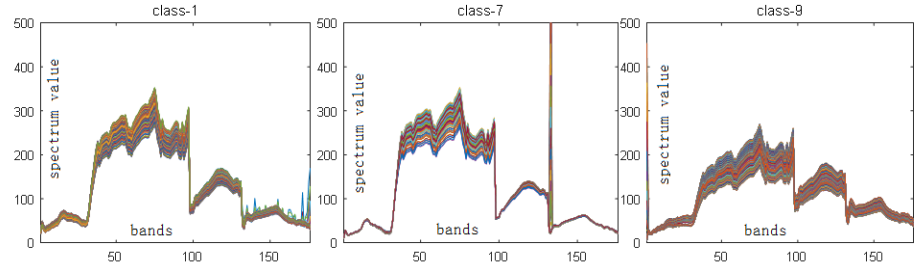


Figure 3. Spectral data of the 3 classes selected from KSC dataset with 176 channels after filtering

First, we need to obtain a guidance image by Principal Component Analysis (PCA). We take the first three principal components as a color guidance image. Given a dataset $D = \{d_1, d_2, \ldots, d_S\}$, we adopt PCA to obtain the following result. Here d_i is the information of the i^{th} band, and S denotes the number of bands.

$$[p_1, p_2, \ldots, p_S] = \text{PCA}(D). \tag{13}$$

So, the guidance image is $G = [p_1, p_2, p_3]$. Then, based on the formula (3), (4), using input image d_1 and guidance image G, we can get the filtering output u_1. By the same way, we can yield all the d_i which construct a new hyperspectral image $U = \{u_1, u_2, \ldots, u_S\}$.

3.2.2 LSTM Model for HSI Classification

The limited number of training samples makes the overfitting a serious problem. To tackle this problem, a regularization strategy based on L2 regularization and dropout is utilized. Meanwhile, HSI samples are in unbalance. The number of samples in some categories is large while in other categories it is small. Previous experiments show that the accuracy of category with the small size is worse than that with the large size. To handle this problem, we implement the weighted cost function, increasing the penalties of small sample misclassification. So, the cross-entropy cost function of LSTM network develops into the following formula:

$$\text{Loss}(T, Y) = -\sum_{i=1}^{n}\sum_{c=1}^{C} w_c * t_{ic} * \log\left(y_{ic} + \frac{\lambda}{2}\|W\|_2\right) \tag{14}$$

where n and C denote the number of samples and categories, respectively. λ is a coefficient of the regularization. We use gradient descent method to learn it. t_{ic} is a true classification label for i^{th} sample in the test set. When the i^{th} sample belongs to the class c, t_{ic} value is one, otherwise, t_{ic} value is zero. y_{ic} is a predicted value of i^{th} sample, which has the same definition with t_{ic}. w_c is the weighted term, obtained

by the following formula:

$$w_c = 1 + \frac{n_{max} - n_c}{n_{max}} \times \theta \tag{15}$$

where n_{max} denotes the number of the largest class, the n_c denotes the number of c-class. θ is a parameter which needs to be learnt.

In addition to the improvements mentioned above, the LSTM model is affected by many other factors, such as normalization, modeling, and optimization function, etc. We investigated the influence of different normalization methods on the HSI classification accuracy. We adopt min-max normalization to normalize the raw data to $[0, 1]$. The normalization formula is $x_{norm} = (x - x_{min})/(x_{max} - x_{min})$, where x denotes the raw data, x_{min} and x_{max} are minimum and maximum values, respectively. Experimental results show that the normalization in the spectral data of a pixel is better than that of the spectral band.

Optimization algorithm is the heart of machine learning, which affects the convergence and optimization of the algorithm. We also found that the Adam is faster and better than the stochastic gradient descent (SGD) optimization method.

Moreover, the network structure of the LSTM model plays an important role in the HSI classification, including input nodes, steps, and hidden layer nodes. We only discuss the influence of input nodes and steps, in this section. Hidden layer nodes are discussed in Section 4. Taking Indian Pines dataset as an example, we study the influence of the number of different input nodes on the classification accuracy. Parameters of the input node and step are set by the following groups, including (2, 100), (5, 40), (8, 25), (10, 20). In which, the product of the number of input nodes and steps is equal to the numbers of bands. The overall accuracy of different modeling methods can be seen from Figure 4. Although (2, 100) looks better at the result, it takes longer training time due to having more time steps. Considering the stability and convergence, we choose (5,40) as the experimental modeling method.

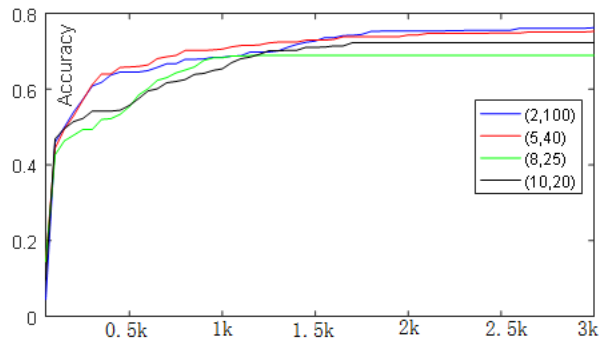


Figure 4. Accuracy of different modeling methods

4 EXPERIMENTS AND RESULTS

All our programs are implemented using Python 2 language and Tensorflow 1.0 library. We implemented all the methods on our PC with 32 GB memory and 8-core CPUs. LSTM methods and AE methods were implemented with a GTX1060 GPU for acceleration. In the following section, we first show the experimental setup, and then describe the experimental process and results in detail.

4.1 Experimental Setup

4.1.1 Datasets

Three hyperspectral data, including Indian Pines and Kennedy Space Center (KSC), are employed to evaluate the effectiveness of the proposed method. The Indian Pines dataset was gathered by Airborne Visible Infrared Imaging Spectrometer (AVIRIS) sensor over the Indian Pines test site in Northwest Indiana. This dataset consists of 145×145 pixels with 200 spectral bands in the wavelength range from 0.4 to 2.5 μm. In this scene, there are 16 categories to be classified, including woods, grass-pasture, etc.

KSC data was also collected by the AVIRIS sensor, acquired in 224 bands of 10 nm with center wavelengths from 0.4 to 2.5 μm. After removing water absorption and low SNR bands, 175 bands were used for the analysis. There are 13 categories to be classified.

Salinas scene was collected by the 224-band AVIRIS sensor over Salinas Valley, and it is characterized by high spatial resolution. It contains 512×217 pixels in all. We also discarded the 20 water absorption bands and selected 200 bands for experiments.

Detailed information of categories and samples is shown in Table 1.

4.1.2 Evaluation Metrics

To evaluate the performance of different HSI classification algorithms, we apply three widely used quality indexes, including the overall accuracy (OA), the average accuracy (AA), and the kappa coefficient (KA). OA shows the number of hyperspectral pixels that are classified correctly, divided by the number of all test samples. AA is the mean of the classification accuracies of all classes. KA is a statistical measurement of agreement, based on the confusion matrix of different classes. If we have a confusion matrix M, where m_{ij} is the element in row i, column j. OA, AA and KA also can be computed by formulas (16), (17), (18), where N and C denote the number and classes of samples.

No.	Indian Pines		KSC		Salinas	
	Categories	**smp**	**Categories**	**smp**	**Categories**	**smp**
C1	Alfalfa	46	Scrub	761	Brocoli_G_W_1	2 009
C2	Corn-N	1 428	Willow swamp	243	Brocoli_G_W_2	3 726
C3	Corn-M	830	CP hammock	256	Fallow	1 976
C4	Corn	237	CP/Oak	252	Fallow_R_P	1 394
C5	Grass-M	483	Slash pine	161	Fallow_smooth	2 678
C6	Grass-T	730	Oak/Broadleaf	229	Stubble	3 959
C7	Grass-P-M	28	Hardwood	105	Celery	3 579
C8	Hay-W	478	Graminoid	431	Grapes_untrained	11 271
C9	Oats	20	Spartina	520	Soil_V_D	6 203
C10	Soybean-N	972	Cattail marsh	404	Corn_S_G_W	3 278
C11	Soybean-M	2 455	Salt marsh	419	Lettuce_R_4wk	1 068
C12	Soybean-C	593	Mud flats	503	Lettuce_R_5wk	1 927
C13	Wheat	205	Water	927	Lettuce_R_6wk	916
C14	Woods	1 265			Lettuce_R_7wk	1 070
C15	Build-G-T-D	386			Vinyard_untrained	7 268
C16	Stone-S-T	93			Vinyard_V_T	1 807
	Total	10 249	Total	5 211	Total	54 129

Table 1. Categories and samples of three datasets

$$OA = \frac{\sum_{i=1}^{C} m_{ii}}{N}, \tag{16}$$

$$AA = \frac{\sum_{i=1}^{C} \frac{m_{ii}}{m_{ii}}}{N}, \tag{17}$$

$$KA = \frac{N \sum_{i=1}^{C} m_{ii} - \sum_{i=1}^{C} m_{i+} m_{+i}}{N^2 - \sum_{i=1}^{C} m_{i+} m_{+i}}. \tag{18}$$

4.2 Influence Factors of the Experiments

4.2.1 Analysis of Sampling Proportion

The proportion of sampling is an important factor affecting the training model. Indian Pines dataset was taken as a case. We exacted samples as training set at the ratio of 5 %, 10 %, 15 %, 20 %, 25 %, and 30 %. We tested three methods including SVM, Autoencoder, and general LSTM, on the above six cases. The experimental results are shown in Figure 5. It illustrates the changes in the accuracy over the proportion from 5 % to 30 %. The increasement of SVM and Autoencoder methods become slow gradually. However, LSTM seems to have a large increase to promote. This shows that LSTM is more dependent on sample size. This is consistent with the fact that deep learning requires a large number of training samples. In order to

accommodate less labeled samples, we choose 10 % as the experimental ratio in this paper.

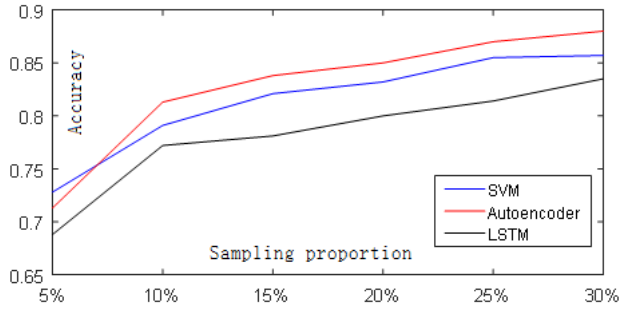


Figure 5. Analysis of sampling proportion in Indian Pines

4.2.2 Analysis on the Number of Hidden Layer Nodes and Iterations

How many iterations does the LSTM model need to reach a stable state? What is the number of hidden nodes to achieve the best classification accuracy? Experiments are done on three datasets. The number of hidden layer nodes are set by 100, 150, 200, 250, and 300, respectively. The results of LSTM with different hidden nodes on three datasets are shown in Figure 6. We can see from the chart, when the number of nodes is 200, the accuracy is the best on Indian Pines dataset. For Kennedy Space Center (KSC) dataset, the number of hidden nodes is 150 for the best result. Salinas dataset choose 200 hidden nodes for our experiment, which is not affected by the number of hidden nodes.

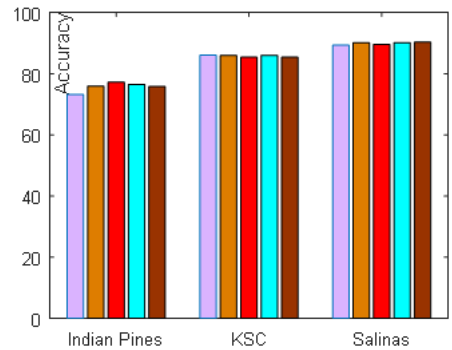


Figure 6. Accuracy of LSTM with different nodes on three datasets

The results of LSTM with different hidden nodes on Indian Pines are shown in Figure 7. It is clear that the accuracy grows sharply for the first one thousand

iterations. They get stable after three thousand iterations. So, we set the number of iterations to 3×103 in the following experiments. Surprisingly, we found that the less nodes the hidden layer has, the slower the initial growth rate is, which is different from our expectations. Another finding is that the network structure with fewer nodes is more prone to underfitting and is more unstable than that with more nodes.

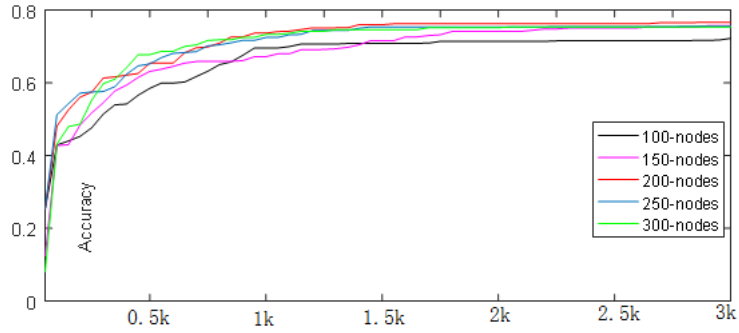


Figure 7. Accuracy of LSTM with different nodes on three datasets

4.3 Experimental Results

To examine the effectiveness of our proposed methods, we do experiments on three widely used datasets. Ten percent of the dataset is taken as training set, and the remaining 90 % as the test set. In this section, the proposed methods are compared with two widely used classification methods, SVM [20] and autoencoder [3], which are typical examples of traditional methods and deep learning methods, respectively.

4.3.1 Experiment on Indian Pines Dataset

Parameter Settings. In this experiment, we use libSVM[2] designed by Lin [6]. The libSVM has two mainly parameters C and g to be set. The C and g are determined by cross validation. C changes from 10^{-2} to 10^4, g ranges from 2^{-1} to 2^4. The parameters of SVM are set by the same way in the following two experiments.

The Autoencoder [3] is based on radius r and regularization parameter ϵ. Radius r is used to express the range of smooth. ϵ is used to control the degree of smooth. We set $r = 3$ and $\epsilon = 0.001$ in this paper. It has the same settings on next two experiments. So, we do not represent them anymore in the following sections.

LSTM, RG-LSTM and GF-LSTM has the same network structure in our experiments. They have three layers, including input layer, hidden layer and output layer,

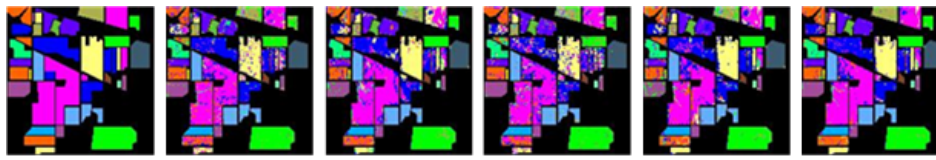
[2] `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

[3] `https://github.com/hantek/deeplearn_hsi`

which are set to 5, 200, and 16, respectively. The step of LSTM is 40. The number of training is set to 5×10^3. RG-LSTM and GF-LSTM have two extra parameters. They are weight coefficient θ and regularization coefficient λ, which are set to 10^{-2} and 10^{-3}, respectively. Besides, radius r and regularization parameter ϵ of guided filter are two important parameters for GF-LSTM. In our previous work [14], the performance of methods is the best when radius r is set to 3 on Indian Pines. In this experiment, we set $r = 3$ and $\epsilon = 10^{-3}$.

Experimental Results. To evaluate our methods, we compared the performance of the methods using the quantitative index of OA, AA and KA. The detailed performance of these methods is shown in Table 2. It can be seen from Table 2 that the OA results of SVM and Autoencoder are better than LSTM by 4 %. LSTM with regularization and weight (RG-LSTM) has improved the classification accuracy of LSTM, which is better than SVM and Autoencoder. Obviously, LSTM with guided filter (GF-LSTM) outperforms all the above methods on all the indexes. Especially in the OA index, GF-LSTM reaches 90.15 %, which is 10 % higher than other methods. Also, overall classification maps of different methods are illustrated in Figure 8. It can be seen from this figure that the proposed methods (RG-LSTM, GF-LSTM) achieve better classification performance than other compared approaches. Besides, the classification results of C9 are poor, especially with Autoencoder. That is because the sample size of C9 is only 20. This explains that sample size has a greater impact on Autoencoder.

For all the methods, the value of OA is higher than that of AA, which means that the class with large samples has a better performance than that with small samples. This problem is more serious in the LSTM. Compared the LSTM, RG-LSTM and GF-LSTM are improved by regularization strategy, which has partly solved the problem of unbalance.



a) Ground truth b) SVM c) Autoencoder d) LSTM e) RG-LSTM f) GF-LSTM

Figure 8. Qualitative results on Indian Pines dataset

4.3.2 Experiment on KSC Dataset

Parameter Settings. For this dataset, there are 4 layers for Autoencoder including two hidden layers. Each hidden layer has 60 nodes. The iteration of pretraining is set to 33×102. Compared to Indian Pines, the hidden layer is reduced, and the iteration is increased. That is because there are too few samples of KSC

	SVM	Autoencoder	LSTM	RG-LSTM	GF-LSTM
C1	83.33	70.52	60.71	76.92	91.30
C2	76.34	74.50	66.53	70.44	84.95
C3	71.73	70.17	73.31	78.14	86.84
C4	48.68	53.77	47.49	50.90	64.63
C5	87.38	85.05	82.34	79.86	90.61
C6	95.06	95.50	89.45	93.05	99.23
C7	86.66	90.00	80.00	68.75	74.71
C8	99.02	95.62	98.75	99.74	100.00
C9	52.63	30.56	52.63	48.57	51.67
C10	75.38	79.41	67.71	74.01	86.00
C11	83.97	69.16	77.56	89.75	93.00
C12	71.12	87.36	69.23	77.33	76.96
C13	93.75	93.07	87.66	88.44	95.59
C14	94.26	94.83	95.22	90.95	98.36
C15	59.77	88.73	55.58	73.23	85.24
C16	86.79	77.48	73.02	91.67	88.00
OA	81.01	81.48	77.02	81.76	90.15
AA	79.12	78.94	74.46	77.99	87.19
KA	78.29	77.29	74.20	79.17	88.24

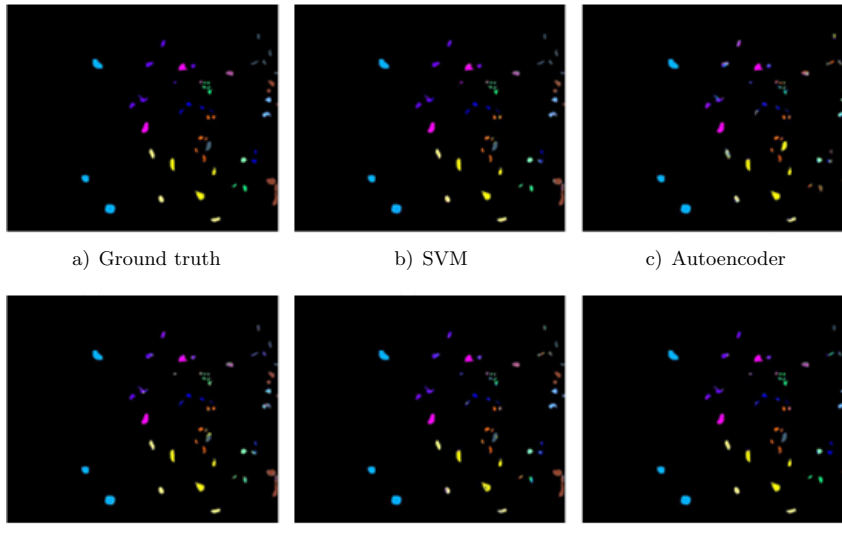
Table 2. Classification accuracy of methods on the Indian Pines dataset (%)

dataset. Other parameters of Autoencoder are the same as the previous experiment.

For LSTM model, we set three layers to train, input layer, hidden layer and output layer. The size of hidden layer is 150, and the input size is 5. The size of time step in the KSC dataset is 35. That is, how many times it carries out the forecast. For the RG-LSTM and GF-LSTM, we set coefficient of the regularization $\lambda = 10^{-4}$, and weight coefficient $\theta = 10^{-2}$. The other settings are the same as the previous experiment.

Experimental Results. The KSC dataset only has about 5 thousand samples. This makes the results different from the previous experiment. The qualitative results are indicated in Figure 9. The results of different methods are shown in Table 3. It can be seen from Table 3, comparing the OA results of SVM, Autoencoder and LSTM, SVM is the best, followed by LSTM, and Autoencoder is the worst, which is 10 % worse than the other two methods. It can be attributed to the small sample size, which makes Autoencoder not fully trained. However, SVM is more suitable for small sample classification. The improved LSTM (RG-LSTM, GF-LSTM) is better than the first three method at OA, AA, and KA. In particular, OA of GF-LSTM increases by 10 % compared with other methods. All these prove that our proposed methods are very effective for HSI classification.

Comparing OA and AA for all the methods, we can see that the gap of Autoencoder is the biggest, which means that Autoencoder is greatly affected by the sample size and sample imbalance. LSTM is less affected by sample imbalance than SVM and Autoencoder. Let us take a look at the classification results of C7, which has the least samples. Autoencoder only has an accuracy about 10.47 %, and LSTM has an accuracy about 57.94 %. With regluarization and weighted cost function, the accuracy of C7 upgrades to 78.79 % by RG-LSTM. Then, with the guided filter, the accuracy is up to 100 % by GF-LSTM, which effectively solves the small size and imbalance of the samples.



a) Ground truth b) SVM c) Autoencoder

d) LSTM e) RG-LSTM f) GF-LSTM

Figure 9. Qualitative results on KSC dataset

4.3.3 Experiment on Salina Dataset

Parameter Settings. For this dataset, there are still 6 layers for Autoencoder as Indian Pines has. Each hidden layer has 60 nodes. Other parameters of Autoencoder are the same as the previous experiment in Indian Pines dataset.

For LSTM model, we set three layers to train, input layer, hidden layer and output layer. The size of hidden layer is 200, and the input size is 5. The size of time step is 40. That is, how many times it carries out the forecast. We set coefficient of the regularization $\lambda = 10^{-4}$, and weight item $\theta = -1 \times 10^{-2}$.

Experimental Results. The last experiment is performed on the Salinas dataset, which is the biggest one we have chosen. The qualitative results are shown in

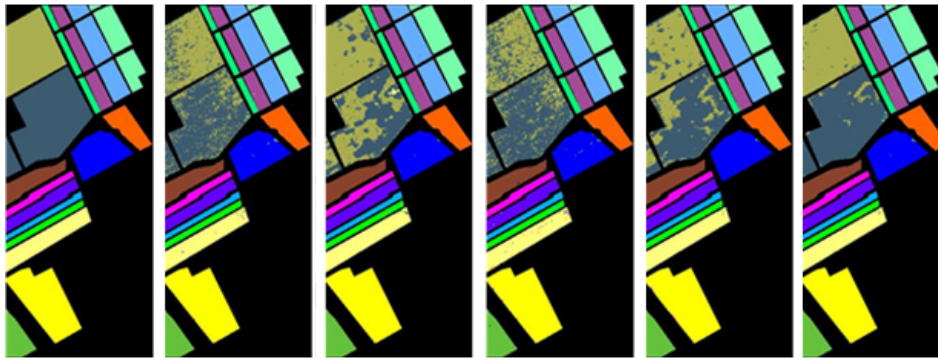
	SVM	Autoencoder	LSTM	RG-LSTM	GF-LSTM
C1	95.59	89.49	90.43	100.00	100.00
C2	73.39	72.94	74.76	71.30	87.20
C3	92.23	27.15	80.08	78.91	98.48
C4	65.47	21.12	61.85	73.10	82.72
C5	76.24	43.38	66.67	72.85	84.03
C6	58.76	24.60	61.39	75.01	97.41
C7	64.58	10.47	57.94	78.79	100.00
C8	79.30	61.92	83.62	90.34	95.58
C9	84.01	78.40	87.91	98.01	100.00
C10	96.87	76.35	83.25	84.87	89.69
C11	85.45	98.90	93.28	94.92	96.41
C12	92.09	58.88	87.78	87.33	90.53
C13	100.0	99.75	98.86	95.87	99.66
OA	87.38	71.86	86.06	88.46	95.48
AA	81.84	58.72	84.37	85.12	93.98
KA	85.87	68.51	79.06	87.08	94.94

Table 3. Classification accuracy of methods on the KSC dataset (%)

Figure 10. It is apparent from this figure that the GF-LSTM obtains the best results, which has the fewest noise points. The detailed results are illustrated in Table 4. All the methods perform well on this dataset. The worst is about 87.4 % by Autoencoder. The OA results of LSTM, RG-LSTM, GF-LSTM are all over 90 %, especially GF-LSTM reaches 98.15 %, outperforming other methods greatly. It can be seen that LSTM with guided filter can significantly improve the classification accuracy.

Comparing AA and OA, we see interesting phenomena that the results of AA are higher than those of OA, which is different from the previous two experiments. This means that when the sample size reaches a certain extent, increasing the sample does not improve the classification accuracy. The gap between AA and OA for Autoencoder is bigger than with other methods. Maybe the number of samples is the main influence factor of Autoencoder. LSTM has better robustness than SVM and Autoencoder. LSTM with regularization (RG-LSTM) can improve the imbalance to some extent. Then, LSTM with guided filter, the results are further improved, whose OA and AA are consistent. Take C8 for example, which is the biggest category, the accuracy is 79.47 % by LSTM. However, the accuracy is up to 88.25 % by RG-LSTM (with regluarization and weighted cost function), and the accuracy reach the highest 97.75 % by GF-LSTM (with guided filter).

From the three experiments we can reach the conclusion that Autoencoder is greatly affected by the sample size. It cannot obtain good results if the size is too large or too small. SVM and LSTM have better robustness. LSTM performs better than SVM on big dataset. RG-LSTM solves the problem of small size and unbalance of samples to a certain extent. GF-LSTM outperforms all the methods at all



a) Ground truth b) SVM c) Autoencoder d) LSTM e) RG-LSTM f) GF-LSTM

Figure 10. Qualitative results on Salinas dataset

the datasets for all indexes. So, GF-LSTM is an effective way to HSI classification.

	SVM	**Autoencoder**	**LSTM**	**RG-LSTM**	**GF-LSTM**
C1	99.94	99.27	99.34	100.00	100.00
C2	99.24	99.75	99.17	100.00	99.85
C3	92.16	95.36	98.49	98.34	99.70
C4	97.22	99.58	99.15	97.77	99.33
C5	98.62	94.67	99.44	99.53	99.32
C6	100.0	99.97	99.86	99.97	99.86
C7	98.93	99.80	99.14	99.20	99.23
C8	80.13	97.13	79.47	88.25	97.75
C9	99.54	99.94	98.95	98.98	99.91
C10	84.39	94.77	96.36	98.67	99.01
C11	85.38	92.03	96.29	97.32	99.59
C12	96.35	100.00	98.45	99.09	99.94
C13	94.97	99.87	95.24	95.00	100.00
C14	94.20	95.63	91.28	96.68	96.31
C15	63.83	18.38	70.67	75.76	93.37
C16	96.15	97.61	94.80	95.38	96.45
OA	88.09	87.40	90.30	91.88	98.15
AA	92.57	92.74	94.76	96.23	98.72
KA	86.65	85.85	89.09	90.89	97.93

Table 4. Classification accuracy of methods on the Salinas dataset (%)

4.3.4 Analysis of Time Cost

In order to compare time cost with the same platform, we reproduced SVM and Autoencoder methods for three datasets. We take the average running time of five times as the time cost, and the results are shown in Table 5. We can see that the running time of LSTM and its improved models are more expensive than SVM and Autoencoder on Indian Pines. For other datasets, LSTM and its improved models are better than SVM on running time cost. Although time cost of Autoencoder is the best, the classification accuracy is the worst. SVM is more sensitive to the size of data, and its running cost will rise rapidly when the amount of data increases. Autoencoder, LSTM, RG-LSTM and GF-LSTM are not particularly sensitive to data size, which is mainly affected by the number of model parameters.

	SVM	**Autoencoder**	**LSTM**	**RG-LSTM**	**GF-LSTM**
Indian Pines	4.011	0.234	4.4548	4.4098	4.4706
KSC	28.507	1.517	1.5554	1.5748	1.5868
Salinas	38.464	0.928	22.5939	23.6033	23.2382

Table 5. Time cost of methods on three datasets

5 CONCLUSION

We proposed a guided filter based LSTM model for HSI classification, under the assumption that the spectral data can be regarded as an ordered sequence. To tackle the problem of unbalance and limited labeled samples, we introduce weighted cost function and regularization strategy. Compared with SVM classifier and Autoencoder model, the proposed model could achieve higher accuracy at all the experimental datasets, with a 10 % samples to train.

Our work is an exploration of using LSTM for HSI classification. The modeling of HSI classification by LSTM can be a useful reference for further research. It is noteworthy that we only take into account the spectral data in this paper, therefore, some spatial-spectral techniques can be employed to improve the LSTM model for classification in the future.

Acknowledgements

This research was funded by the Science and Technology Project for the Universities of Shandong Province (grant No. J18KB171), Open Research Fund of Shandong Provincial Key Laboratory of Infectious Disease Control and Prevention, Shandong Center for Disease Control and Prevention (grant No. 2017KEYLAB01), Shandong Women's University High Level Scientific Research Project Cultivation Fund (grants No. 2020GSPGJ08 and 2019GSPGJ07), Discipline Talent Team Cultivation Program of Shandong Women's University (grant No. 1904).

REFERENCES

[1] Bioucas-Dias, J. M.—Nascimento, J. M. P.: Hyperspectral Subspace Identification. IEEE Transactions on Geoscience and Remote Sensing, Vol. 46, 2008, No. 8, pp. 2435–2445, doi: 10.1109/tgrs.2008.918089.

[2] Chen, Y.—Jiang, H.—Li, C.—Jia, X.—Ghamisi, P.: Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks. IEEE Transactions on Geoscience and Remote Sensing, Vol. 54, 2016, No. 10, pp. 6232–6251, doi: 10.1109/tgrs.2016.2584107.

[3] Chen, Y.—Lin, Z.—Zhao, X.—Wang, G.—Gu, Y.: Deep Learning-Based Classification of Hyperspectral Data. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, Vol. 7, 2014, No. 6, pp. 2094–2107, doi: 10.1109/jstars.2014.2329330.

[4] Chen, Y.—Nasrabadi, N. M.—Tran, T. D.: Hyperspectral Image Classification via Kernel Sparse Representation. IEEE Transactions on Geoscience and Remote Sensing, Vol. 51, 2013, No. 1, pp. 217–231, doi: 10.1109/TGRS.2012.2201730.

[5] Chen, Y.—Zhao, X.—Jia, X.: Spectral-Spatial Classification of Hyperspectral Data Based on Deep Belief Network. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, Vol. 8, 2015, No. 6, pp. 2381–2392, doi: 10.1109/TGRS.2012.2201730.

[6] Chang, C. C.—Lin, C. J.: LIBSVM: A Library for Support Vector Machines. ACM Transactions on Intelligent Systems and Technology, Vol. 2, 2011, No. 3, Art. No. 27, doi: 10.1145/1961189.1961199.

[7] Dalponte, M.—Ørka, H. O.—Gobakken, T.—Gianelle, D.—Næsset, E.: Tree Species Classification in Boreal Forests with Hyperspectral Data. IEEE Transactions on Geoscience and Remote Sensing, Vol. 51, 2013, No. 5, pp. 2632–2645, doi: 10.1109/TGRS.2012.2216272.

[8] Fauvel, M.—Chanussot, J.—Benediktsson, J. A.: Kernel Principal Component Analysis for the Classification of Hyperspectral Remote Sensing Data over Urban Areas. EURASIP Journal on Advances in Signal Processing, Vol. 1, 2009, Art. No. 783194, doi: 10.1155/2009/783194.

[9] Feng, J.—Jiao, L. C.—Zhang, X.—Sun, T.: Hyperspectral Band Selection Based on Trivariate Mutual Information and Clonal Selection. IEEE Transactions on Geoscience and Remote Sensing, Vol. 52, 2014, No. 7, pp. 4092–4105, doi: 10.1109/tgrs.2013.2279591.

[10] Hastie, T.—Tibshirani, R.—Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Vol. 1, Springer Series in Statistics, New York, 2001.

[11] Graves, A.: Supervised Sequence Labelling. Supervised Sequence Labelling with Recurrent Neural Networks. Springer, Berlin, Heidelberg, Studies in Computational Intelligence, Vol. 385, 2012, pp. 5–13, doi: 10.1007/978-3-642-24797-2_2.

[12] Graves, A.—Mohamed, A.-r.—Hinton, G.: Speech Recognition with Deep Recurrent Neural Networks. 2013 IEEE International Conference on Acoustics,

Speech and Signal Processing, Vancouver, BC, Canada, 2013, pp. 6645–6649, doi: 10.1109/icassp.2013.6638947.

[13] Gregor, K.—Danihelka, I.—Graves, A.—Rezende, D. J.—Wierstra, D.: DRAW: A Recurrent Neural Network for Image Generation. arXiv preprint arXiv:1502.04623, 2015.

[14] Guo, Y.—Cao, H.—Han, S.—Sun, Y.—Bai, Y.: Spectral-Spatial Hyperspectral Image Classification with k-Nearest Neighbor and Guided Filter. IEEE Access, Vol. 6, 2018, pp. 18582–18591, doi: 10.1109/access.2018.2820043.

[15] He, K.—Sun, J.—Tang, X.: Guided Image Filtering. In: Daniilidis, K., Maragos, P., Paragios, N. (Eds.): Computer Vision – ECCV 2010. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6311, 2010, pp. 1–14, doi: 10.1007/978-3-642-15549-9_1.

[16] Hinton, G. E.—Salakhutdinov, R. R.: Reducing the Dimensionality of Data with Neural Networks. Science, Vol. 313, 2006, No. 5786, pp. 504–507, doi: 10.1126/science.1127647.

[17] Hu, W.—Huang, Y.—Wei, L.—Zhang, F.—Li, H.: Deep Convolutional Neural Networks for Hyperspectral Image Classification. Journal of Sensors, Vol. 2015, 2015, Art. No. 258619, doi: 10.1155/2015/258619.

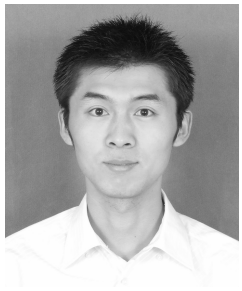
[18] Hughes, G.: On the Mean Accuracy of Statistical Pattern Recognizers. IEEE Transactions on Information Theory, Vol. 14, 1968, No. 1, pp. 55–63, doi: 10.1109/tit.1968.1054102.

[19] Ma, L.—Crawford, M. M.—Tian, J.: Local Manifold Learning-Based k-Nearest-Neighbor for Hyperspectral Image Classification. IEEE Transactions on Geoscience and Remote Sensing, Vol. 48, 2010, No. 11, pp. 4099–4109, doi: 10.1109/tgrs.2010.2055876.

[20] Melgani, F.—Bruzzone, L.: Classification of Hyperspectral Remote Sensing Images with Support Vector Machines. IEEE Transactions on Geoscience and Remote Sensing, Vol. 42, 2004, No. 8, pp. 1778–1790, doi: 10.1109/TGRS.2004.831865.

[21] Mikolov, T.—Karafiát, M.—Burget, L.—Černocký, J.—Khudanpur, S.: Recurrent Neural Network Based Language Model. Eleventh Annual Conference of the International Speech Communication Association (INTERSPEECH 2010), Makuhari, Chiba, Japan, 2010, pp. 1045–1048.

[22] Wang, J.—Chang, C.-I.: Independent Component Analysis-Based Dimensionality Reduction with Applications in Hyperspectral Image Analysis. IEEE Transactions on Geoscience and Remote Sensing, Vol. 44, 2006, No. 6, pp. 1586–1600, doi: 10.1109/tgrs.2005.863297.



Yanhui Guo received his B.Sc. degree in information management and information system from the Xi'an University of Finance and Economics, Xi'an, China in 2006, and his M.Sc. degree in computer software and theory from the Shaanxi Normal University, Xi'an, China in 2009, then he received his Ph.D. degree with the School of Computer Science of Shaanxi Normal University in 2020. Since 2009, he has been with the School of Information Technology, Shandong Women's University, Jinan, China, where he is currently Professor. His research interests include computer vision and machine learning.

Fuli Qu received her Master's degree in computational mathematics at Shandong University. She is Assistant Professor at the Institute of Data Science and Computing. Her research focuses on numerical methods of differential equations. She published her thesis at applied mathematics and mechanics.

Zhenmei Yu received her M.Sc. degree from the School of Management Science and Engineering at Shandong Normal University. She is now Professor at the School of Data and Computer Science, Shandong Women's University. Her main research interests include machine learning and artificial intelligence.

Qian Yu is currently Associate Professor in the School of Data and Computer Science, Shandong Women's University, China. She received her Ph.D. degree from the Department of Computer Science of Nanjing University in 2020. Her research interests include computer vision and medical image analysis.

Computing and Informatics, Vol. 39, 2020, 994–1021, doi: 10.31577/cai_2020_5_994

DETERMINING THE RELATIVE IMPORTANCE OF PERSONALITY TRAITS IN INFLUENCING SOFTWARE QUALITY AND TEAM PRODUCTIVITY

Nosheen Qamar

Department of Computer Science and Information Technology
University of Lahore
Lahore, Pakistan
&
Department of Computer Science
National University of Computer and Emerging Sciences
Lahore, Pakistan
e-mail: nosheen.qamar@cs.uol.edu.pk

Ali Afzal Malik

Department of Computer Science
National University of Computer and Emerging Sciences
Lahore, Pakistan
e-mail: ali.afzal@nu.edu.pk

Abstract. Software projects are almost always team efforts and successful projects involve well-formed and well-composed teams. Past studies have revealed that personality contributes to effective team composition and, therefore, project success. Yet despite its importance, only a couple of empirical studies have quantitatively evaluated the impact of personality on software quality and team productivity. Our previous study was an effort in this direction. In that study, we proposed a metric called Team Homogeneity Index and evaluated its impact on software quality and team productivity for two phases (implementation and testing) of the software development life cycle. This study is a continuation of our previous work. In this study, we replicate our experiment on three different phases of software development life cycle (i.e. analysis and design, implementation, and testing). We also determine the weights for all five personality traits using input from the industry and propose

an improved version of Team Homogeneity Index called Weighted Team Homogeneity Index. Finally, we conduct a comparative analysis of Team Homogeneity Index and Weighted Team Homogeneity Index to determine whether weights assigned to personality traits make any difference. Our findings reveal that weights do make a difference and Weighted Team Homogeneity Index is more strongly correlated than Team Homogeneity Index for almost all of the teams, especially those composed of practitioners, in the three different phases of Software Development Life Cycle.

Keywords: Personality traits, social aspects of software engineering, software developer, software quality, team homogeneity, team productivity

1 INTRODUCTION

The modern society progressively demands quality and productivity in all aspects of life. In every field, including software development, people are ready to adopt new approaches to improve product quality. Software quality refers to the degree to which all attributes of a software system appropriately fulfill its requirements [15]. Software development productivity, on the other hand, is defined as the functional value of developed software in relation to the cost and labor consumed while developing that software [43].

As reported by Pressman and Maxim [34], software is designed and developed by the people for the people and maintains a connection between them. People are considered as an important factor in influencing the success or failure of software projects [6]. According to DeMarco and Lister, software projects fail mostly because of incompetent teams [14].

Software teams depend on communication, negotiation, collaboration, and administrative skills to make a project successful [6]. Therefore, appropriate composition of teams is crucial. However, despite the significance of team composition most of the previous studies have focused on technical aspects instead of human, personality, and psychological factors [24].

Only some researchers [6, 14, 24] have investigated the importance of human factors in software development. Their investigations have focused on team members and tasks performed by them to achieve a successful project. Past research [4, 9, 11] has demonstrated that a positive relation exists between personality and team performance.

Personality is defined as attributes and characteristics which make an individual unique [30]. It can be studied by using various popular personality models, such as Myers-Briggs Type Indicator (MBTI) [28], Keirsey Temperament Sorter (KTS) [22], and Five-Factor Model (FFM) also called "Big Five" model [26]. The first two models revolve around personality types while the third is based on personality traits (i.e. human characteristics in diverse dimensions [28]). MBTI, a commonly used

model, employs four components of a character, i.e. Sensing/Instinct, Extraversion/Introversion, Judging/Perceiving, and Thinking/Feeling, while KTS uses four temperaments namely Idealist, Guardian, Artisan, and Rational.

FFM quantifies character using the five personality traits described below [26]:

Openness: An individual with this trait is keen to explore new things and loves creative ideas. In addition, he or she has the power to handle diverse situations in the right manner.

Conscientiousness: Individuals with this trait think and plan everything conscientiously. They prefer to predict the situation and plan accordingly. Once they plan their milestones, they stick to them with great effort and responsibility.

Extraversion: People with this trait are lively, energetic, cheerful, assertive, social, and have extraordinary communication skills.

Agreeableness: People with this trait are trustworthy and have a warm frame of mind. They are always ready to assist others and are of a kind heart.

Neuroticism: Individuals with this trait are inclined to get discouraged, stressed, irritated, and disappointed more frequently.

A number of researchers have conducted qualitative analyses to evaluate the impact of personality on project quality and team productivity [4, 6, 9, 11, 14, 24, 34, 46]. To the best of our knowledge, only two studies have quantitatively measured the team personality and that too by using a measure of central tendency (i.e. mean) which is not always considered a good representative of a dataset. Our previous research [35] was the first study that quantified the notion of team homogeneity using a measure of spread thereby taking data variation and dispersion into account. In that research, we proposed a new metric called Team Homogeneity Index (THI). We also conducted experiments to evaluate the impact of THI on software quality and team productivity during the implementation and testing phases of SDLC.

The aim of this study is to extend our previous research by replicating our experiment with more students in addition to engaging practitioners during software analysis and design, implementation, and testing phases of SDLC to check whether our previous results are generalizable. Furthermore, we have introduced an improved version of THI called Weighted Team Homogeneity Index (WTHI) which uses weights for each of the five personality traits. This improvement is in line with previous research [4, 5, 8, 29] which shows that all traits are not equally important for the software industry. These weights have been obtained by conducting a survey of the Pakistani software industry. Moreover, we have performed a comparative analysis of THI and WTHI with respect to their influence on software quality and team productivity.

Our hypotheses for this research are:

HA0: THI will have no or rather negative relationship with team productivity.

HA1: Teams with higher values of THI will be more productive.

HB0: THI will have no or negative relationship with the quality of software.

HB1: Teams with higher values of THI will produce better quality software.

HC0: WTHI will have no or negative relationship with team productivity.

HC1: Teams with higher values of WTHI will be more productive.

HD0: WTHI will have no or negative relationship with the quality of software.

HD1: Teams with higher values of WTHI will produce better quality software.

HE0: WTHI will be no better in predicting the productivity of teams than THI.

HE1: WTHI will be better in predicting productivity of teams than THI.

HF0: WTHI will be no better in predicting the quality of software than THI.

HF1: WTHI will be better in predicting the quality of software than THI.

The rest of this paper is structured as follows. Section 2 provides a brief summary of related work. The process of determination of weights and calculation of WTHI is described in Section 3. Section 4 describes the assessment criteria while Section 5 provides the details of our experiment. Section 6 discusses the results achieved and Section 7 highlights the threats to the validity of our research. Major conclusions and directions for future work are summarized in Section 8.

2 RELATED WORK

The influence of personality on software teams has been discussed several times in various studies conducted in both industrial and academic environments. In these studies, the main focus was on understanding how much personality influences the performance of a team [4, 7, 9, 11, 13, 24]. The process of assembling a team on the basis of the attributes of its members is known as team composition [25]. These attributes include their experience, demographics, expertise, and other factors regarding their individual personalities [25].

It has been perceived that some individuals can be more productive than others [27, 43]. Similarly, some team members contribute more to the quality of the product as compared to other members. Therefore, it makes a lot of sense to focus on how to create viable teams which can proficiently and effectively develop high quality products [37, 46].

Findings from existing studies show that team composition considerably affects the performance of a team [8]. Some studies recommend that a team should comprise of different personality types to enhance the team's performance [20, 32]. Other studies suggest that a team composed of the same personality types performs better [23, 29].

Different studies have used different personality models to assess the personality of software development teams and have assessed the impact of personality on software quality and team productivity. For example, Rutherfoord [38], Golra and Lam [17], and Sfetsos et al. [42] used KTS as personality assessment tool. Their

results indicated that diverse teams perform better because they communicated and collaborated more as compared to homogeneous teams.

Capretz [10] conducted a survey involving 100 professionals using MBTI personality model. Results revealed that people having diverse skills and personalities form better teams. Peslak [31] also conducted a survey with 55 students. He concluded that extraversion, thinking, and judging personality characteristics positively correlate with project success. Furhtermore, Karn et al. [21], Choi et al. [12], and Poonam and Yasser [33] also used MBTI for personality assessment. Results of [21] revealed that homogeneous teams proved to be highly cohesive and, hence, performed well whereas the results of [12] indicated that the teams with diverse groups were more productive than alike and apposite pair groups. [33] indicated that the performance of pairs working remotely was affected by personality traits.

Walle and Hannay [45] conducted a survey with 88 professionals using FFM personality model. Their results indicated that personality attributes contribute to the collaboration of a team. Later, Salleh et al. [39, 41, 40] used FFM as personality assessment tool in their experiments with 453 students. Their findings showed that Conscientiousness and Neuroticism traits have no significant impact on students' performance whereas a positive correlation was found between openness and teams' performance. Yilmaz et al. [47] also conducted a survey involving 216 professionals. They concluded that practitioners were found to be more extrovert and effective teams were observed to be emotionally more stable.

Acuña et al. [4] calculated team personality by taking the average of personality score of each team member. A positive correlation between the quality of software product and extraversion was found. In 2015, Acuña et al. [5] repeated the same experiment at a larger scale and the results of this experiment were the same as of the first experiment [4]. Furthermore, a positive correlation between product quality and high participative safety and task orientation climate perceptions was observed.

Earlier, we conducted an empirical study [35] to assess the impact of our newly proposed metric, THI, on software quality and team productivity. Our results revealed that THI has a positive impact on different quality factors and team productivity for software implementation and testing phases of SDLC.

3 DETERMINATION OF WEIGHTS AND CALCULATION OF WTHI

As shown in Figure 1, determination of weights is now the first step of our research methodology. To determine the weights of the five personality traits, we conducted a survey [2] of the Pakistani software industry by engaging all members of the Pakistan Software Houses Association (P@sha). A total of 107 professionals belonging to 49 different companies participated in our survey.

Figure 2 shows detailed information about the work experience, rank, and role of these respondents. It also shows the commonly used software development process in the respondent's companies. Figure 2 a) shows that around 50 % of respondents had

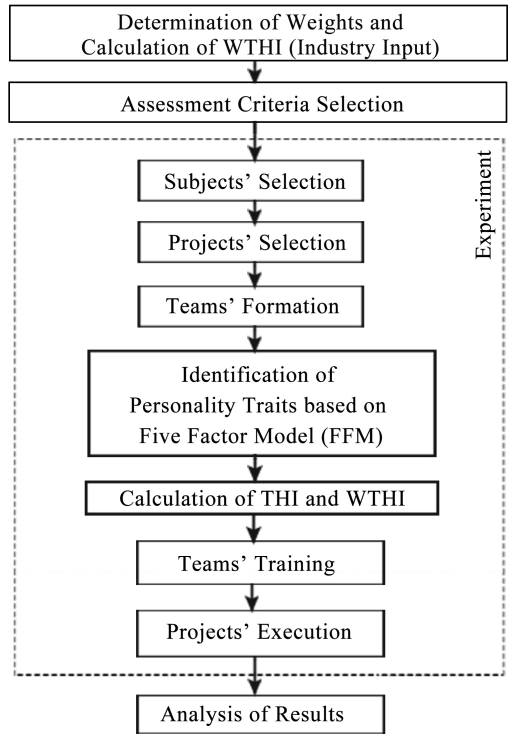
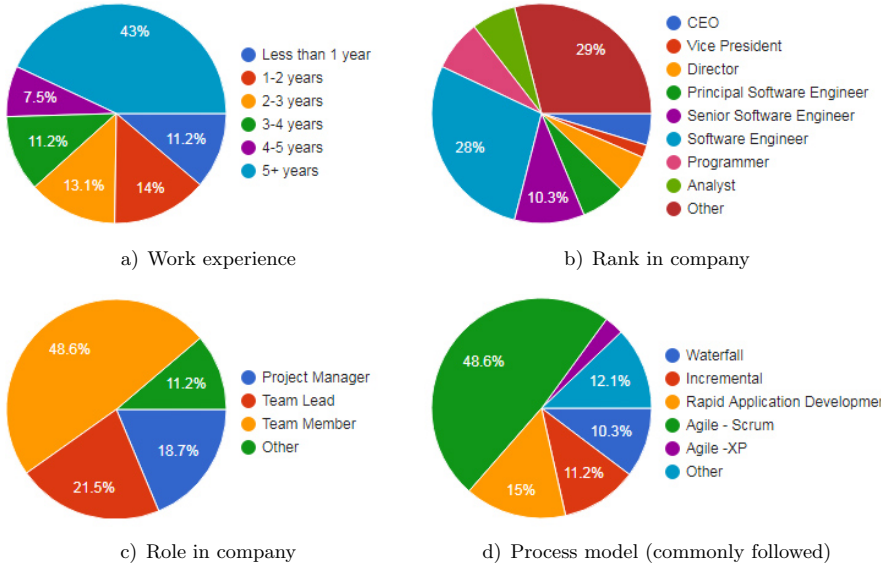


Figure 1. Research methodology (adapted from [35])

four or more years of work experience. This indicates that the input was given by experienced professionals. Figure 2 b) reveals that more than 10 % of the respondents held top management positions like CEO, vice president, and directors in their respective companies. Around 40 % of respondents were playing the role of a team lead or project manager (see Figure 2 c)). It is clear from Figure 2 d), that almost half of the companies represented by these respondents were using Scrum.

Figure 3 shows the frequency of different weights on a scale of 1 to 5 with 1 being least important and 5 being most important provided by respondents for each of the five personality traits. The final value of the weight for each personality trait was calculated by using the arithmetic mean of the weights provided by the respondents. These final values are shown in Figure 4. This figure shows that Openness was considered the most important while Neuroticism was considered the least important personality trait in influencing the software quality and team productivity.

The quantification of THI is a six-step process. The quantification of WTHI also follows the same steps except that it uses weights obtained from the industry (see Figure 4). The complete quantification process is described in Table 1. Figure 5



a) Work experience b) Rank in company

c) Role in company d) Process model (commonly followed)

Figure 2. Survey respondents' information

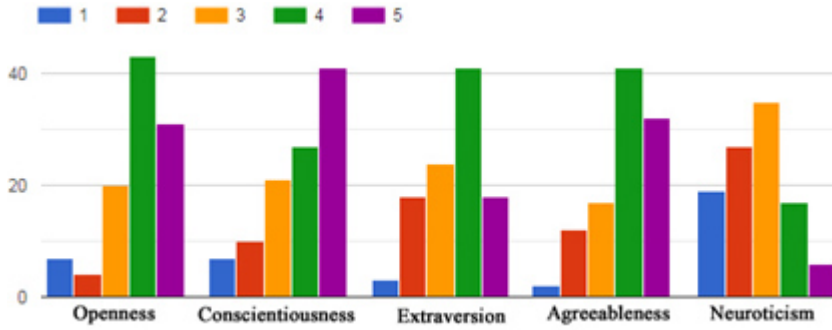


Figure 3. Frequency of weights for each personality trait

shows a detailed worked-out example of calculating WTHI for a five-members team using these steps.

4 ASSESSMENT CRITERIA

4.1 Assessment Criteria for Analysis and Design Phase

Table 2 lists the criteria (adapted from [19]) used to evaluate the quality of the analysis and design models produced by the teams. The quality of these models

Sr.	Step	Description
1	Identification and Normalization of Personality Scores	Identification: 50-item test from IPIP [16] Normalization: Min-Max
2	Calculation of Individual Heterogeneity (find the heterogeneity between the score of one team member and the scores of all other team members)	$H = \lvert p_k - q_k \rvert$ where H = Heterogeneity p_k = one team member q_k = other team member
3	Calculation of Overall Heterogeneity (divide the sum of all the weighted heterogeneity by the no. of traits) For WTHI, actual weights (Figure 4) were used For THI, a weight of 1 for all the traits was used	$OH = 1/n \left(\sum_{k=1}^{n} w_k \lvert p_k - q_k \rvert \right)$ where OH = Overall Heterogeneity n = total number of personality traits w_k = weight of the k^{th} trait
4	Calculation of Mean (divide the sum of the overall heterogeneity for all the member-pairs with the total number of member-pairs)	$\text{Mean} = (x_1 + x_2 + x_3 + \cdots + x_n)/m$ where $x_1, x_2, \ldots, x_n = OH$ of all member-pairs m = all member pairs (i.e. 10)
5	Calculation of Mean Absolute Error (MAE) (subtract the overall heterogeneity of each member-pair from the mean)	$MAE = 1/m \sum_{k=1}^{m} \lvert \text{mean} - x_k \rvert$ where MAE = Mean Absolute Error $x_1, x_2, \ldots, x_n = OH$ of all member-pairs m = all member pairs (i.e. 10)
6	Calculation of (W)THI (subtract the MAE from 1) Homogeneity lies between 0 to 1 where 0 indicates no homogeneity and 1 indicates that the team is 100 % homogeneous	$(W)THI = 1 - MAE$

Table 1. Quantification process of THI and WTHI

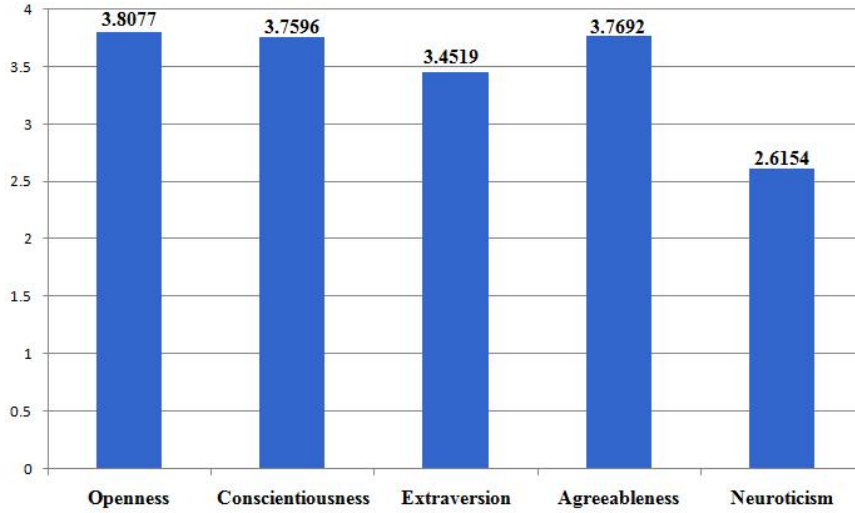


Figure 4. Final weights for each personality trait

was assessed using 24 different factors (grouped under four different categories). The productivity of teams was evaluated using the percentage completeness (using 6 factors) of the analysis and design documents.

4.2 Assessment Criteria for Implementation Phase

Four different quality factors were used to assess the quality of implemented projects i.e. Defect Density, Weighted Sum of Bugs (WSB), Maintainability Index (MI), and Cyclomatic Complexity (CC). The productivity of teams during this phase was determined by taking the ratio of project completeness (weighted sum of implemented features expressed as a percentage and effort taken to complete the project. The detailed assessment criteria for implementation phase are provided in [35].

4.3 Assessment Criteria for Testing Phase

The quality of testing was assessed by looking at Defects Uncovered (number of failed test cases), Architectural Coverage (features tested divided by total features of the project), and Test Case Conformity ((correct test case attributes/total test case attributes) $*$ 100) (adapted from [36]). The detailed criteria are given in [35].

5 EXPERIMENT

In order to assess the utility of THI and WTHI in predicting software quality and team productivity, we performed a formal experiment in which software quality and

Step 1 (a). Team Members' Scores Obtained From FFM Test

Sr#	P.T. / T.M.	Ope	Con	Ext	Agr	Neu
1	Member 1	35	22	24	31	28
2	Member 2	20	27	37	20	31
3	Member 3	18	19	21	19	38
4	Member 4	25	21	30	25	46
5	Member 5	30	18	32	26	23

=>

Step 1 (b). Team Members' Scores After Normalization

Sr#	P.T. / T.M.	Ope	Con	Ext	Agr	Neu
1	Member 1	1.000	0.444	0.188	1.000	0.217
2	Member 2	0.118	1.000	1.000	0.083	0.348
3	Member 3	0.000	0.111	0.000	0.000	0.652
4	Member 4	0.412	0.333	0.563	0.500	1.000
5	Member 5	0.706	0.000	0.688	0.583	0.000

*Ope= Openness, Con= Conscientiousness, Ext=Extraversion, Agr=Agreeableness, Neu=Neuroticism, T.M.= Team Member, P.T.= Personality Trait

Step 2 (a). Heterogeneity w.r.t. Openness

0.882			
1.000	0.118		
0.588	0.294	0.412	
0.294	0.588	0.706	0.294

Step 2 (b). Heterogeneity w.r.t. Concientiousness

0.556			
0.333	0.889		
0.111	0.667	0.222	
0.444	1.000	0.111	0.333

Step 2 (c). Heterogeneity w.r.t. Extraversion

0.813			
0.188	1.000		
0.375	0.438	0.563	
0.500	0.313	0.688	0.125

Step 2 (d). Heterogeneity w.r.t. Agreeableness

0.917			
1.000	0.083		
0.500	0.417	0.500	
0.417	0.500	0.583	0.083

Step 2 (e). Heterogeneity w.r.t. Neuroticism

0.130			
0.435	0.304		
0.783	0.652	0.348	
0.217	0.348	0.652	1.000

=((3.8077*0.882)+(3.7596*0.556)+(3.4519*0.813)+(3.7692*0.917)+(2.6154*0.130))/5

Step 3. Overall Heterogeneity

2.410			
2.123	**1.670**		
1.577	**1.683**	**1.428**	
1.331	**1.974**	**1.877**	**1.147**

Weight of Openness trait = 3.8077
Weight of Conscientiousness trait = 3.7596
Weight of Extraversion trait = 3.4519
Weight of Agreeableness trait = 3.7692
Weight of Neuroticism trait = 2.6154

Step 4. Mean Heterogeneity = (2.410+2.123+1.577+1.331+1.670+1.683+1.974+1.428+1.877+1.147)/10 = 1.722

Step 5. MAE = (|1.722-2.410|+|1.722-2.123|+|1.722-1.577|+|1.722-1.331|+|1.722-1.670|+|1.722-1.683|+|1.722-1.974|+|1.722-1.428|+|1.722-1.877|+|1.722-1.147|)/10 = 0.299

Step 6. WTHI = 1-0.299 = 0.701

Figure 5. WTHI calculation example (adapted from [35])

team productivity were dependent variables while THI and WTHI were independent variables. Different steps of our experiment (shown in Figure 1) are described below.

5.1 Subjects' Selection

This experiment was carried out in both academic and industrial environments. From the industry, 35 professionals (33 male and 2 female) participated in this experiment while from the academia, a total of 215 BS (Computer Science) students (197 male and 18 female) participated. This experiment was carried out in three different phases of the SDLC, i.e. analysis and design, implementation, and testing. 50 students studying the "Software Engineering" course participated in the analysis and design phase, 90 students enrolled in the "Web Engineering" course participated in the implementation phase, and 75 students registered in the "Software Testing" course took part in the testing phase. The professionals' teams worked on all of the above three phases of SDLC.

Categories	Factors
Completeness	Functionality Model Abstraction Missing Operations Strange Relationships Missing Classes and Attributes Functions Parameters and their Data Types
Understandability	Complexity Easy to Read Class, Attributes, and Operations Names Relationships Names Number of Classes, Operations, and Attributes Extra Information
Correctness	Correctness of Data Flow and Layout Correctness of Entities Correctness of Relationships Correctness of Operations Correctness of Sequences Correctness of Classes Correctness of Attributes Conformance to the Standards
Layout	Good Class Name Neat or Chaotic Structure Classes with Similar Size Classes Hierarchy and Alignment Distance between Classes Line Style (Overlapping, Crossing, and Bend)
Use of Relationships at Appropriate Places	Number of Associations Number of Aggregations Number of Generalizations Number of Compositions

Table 2. Analysis and design teams' quality assessment criteria

5.2 Projects' Selection

For professionals, a relatively large project (Online Job Portal) was selected. For students, a comparatively small project (My Shop) was chosen. These same projects were used for all three phases of SDLC.

5.3 Teams' Formation

There were five members in each team of both professionals and students. To keep professionals' teams similar, it was ensured that each team's average experience was between 3–4 years. It was also made sure that educational qualification of every

professional was a bachelors degree in computer science and his/her age was between 24 and 30 years.

To keep students' teams similar, we formed their teams using three buckets of CGPA (i.e. bucket A: 3.00–4.00, bucket B: 2.50–3.00, and bucket C: 2.00–2.50) in such a way that each team consisted of one member from bucket A, two members from bucket B, and two from bucket C. Besides this, it was ensured that no more than one female was assigned to a team and no more than one member in a team had some previous experience related to software development (e.g. internship, freelancing etc.).

5.4 Identification of Personality Traits

Personality traits of all individuals were identified [1] using a 50-item five-factor personality test from the International Personality Item Pool (IPIP) [16].

5.5 Calculation of THI and WTHI

Data obtained in the previous step was used and processed further to compute THI and WTHI. The calculation process of THI and WTHI is explained in detail in [35] and Section 3, respectively.

5.6 Teams' Training

A total of ten training sessions were conducted. There were two types of sessions: a 30-minutes session before the personality assessment test and a 3-hours session before the start of the actual experiment. Two of these ten training sessions were conducted for professionals (one 30-minutes session and one 3-hours training session). Four training sessions were conducted for implementation teams (two 30-minutes sessions and two 3-hours sessions) as there were 90 subjects and to maintain the quality of training sessions they were divided into two groups. Similarly, two sessions (one 30-minutes and one 3-hours) were conducted for each of the analysis and design and testing teams. The agenda of training sessions included explanation of personality test, discussion on Software Requirements Specification (SRS) document provided to teams, guidelines related to experiment, and tutorial of the Time Keeper tool [3] used to record time.

5.7 Projects' Execution

After the successful completion of all the training sessions, the Online Job Portal project was handed over to professionals' teams for performing analysis and design, implementation, and testing. The My Shop project was given to "Software Engineering" students for analysis and design, "Web Engineering" students for implementation, and "Software Testing" students for testing. The maximum time given

to professionals for all three phases combined was 8 weeks (each professional was required to work maximum of 5–6 hours per week). An agreement was reached with the associated software houses that they will have the ownership of the developed projects and we will be allowed to use the project data/documents for research. In the case of students, the time given for analysis and design and implementation of the project was two and four weeks, respectively. Two working days were given for testing.

6 RESULTS AND DISCUSSION

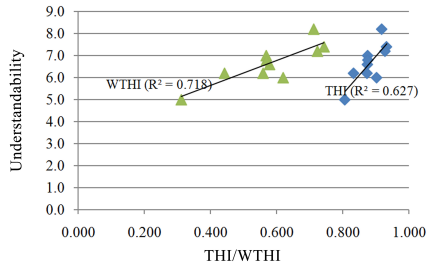
6.1 Analysis and Design Phase

Table 3 shows the details related to the analysis and design models created by the teams. Column 5 contains the scores of models' (class, entity-relationship, data flow, activity, and sequence) overall understandability on a scale of 1 to 10, where 1 means the model is very difficult to understand and 10 means the model is very easy to understand. Overall understandability is calculated by taking the average of the models' individual understandability scores. Column 6 contains the models' overall correctness scores where 1 denotes least correct and 10 represents most correct. Overall correctness is calculated by averaging the models' individual correctness scores. Column 7 contains the models' overall layout scores (obtained by averaging the individual scores). Column 8 contains the average number of relationships (generalization, association, aggregation, and composition) used at appropriate places. Productivity of analysis and design teams appears in the last column.

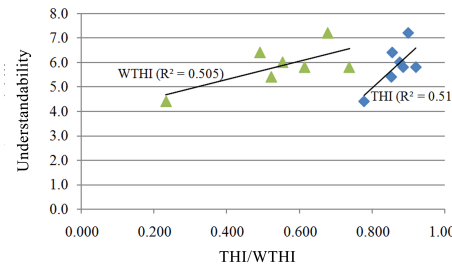
Figure 6 shows the impact of THI and WTHI on analysis and design teams' models' quality. Figures 6 a) and 6 b) depict the relationship between THI or WTHI and understandability for both students' and professionals' teams. It can be seen that the understandability scores get higher with an increase in the values of THI and WTHI. This supports our hypotheses HB1 and HD1. WTHI seems to have a stronger impact on understandability for students' teams (supporting HF1) but, for professionals' teams, the impact of WTHI on understandability is not relatively strong (HF0 cannot be rejected).

Figures 6 c), 6 d), 6 e) and 6 f) compare the impact of THI and WTHI on students' and professionals' models' correctness scores, layout scores, and appropriately used relationships, respectively. The upward slopes of all trend-lines indicate that correctness, layout and appropriately used relationships have a positive correlation with THI and WTHI supporting our HB1 and HD1 hypotheses. Since WTHI has a stronger impact on correctness, layout, and relationships (as compared to THI), HF1 is also supported for both students' and professionals' teams.

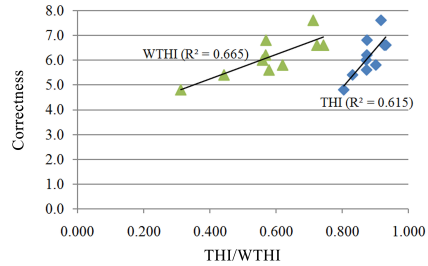
Figure 7 a) displays scatter plots with trend-lines that depict the impact of THI and WTHI on the productivity of students' teams. Figure 7 b) shows the same for professionals' teams. The trend-lines clearly indicate that the teams with greater THI and WTHI values were more productive. This supports our HA1 and HC1 hypotheses. It is worth noting that, for professionals, WTHI has a stronger



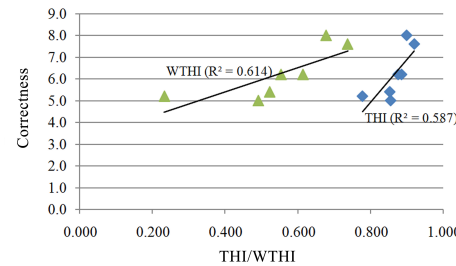
a) Impact of THI and WTHI on understandability (students)



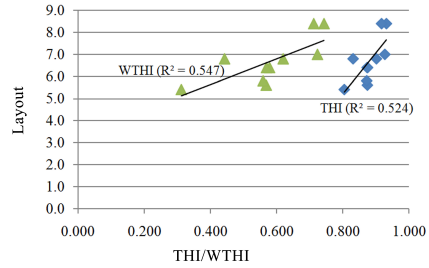
b) Impact of THI and WTHI on understandability (professionals)



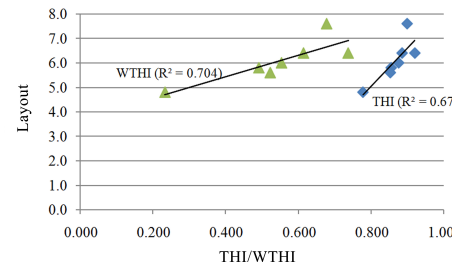
c) Impact of THI and WTHI on correctness (students)



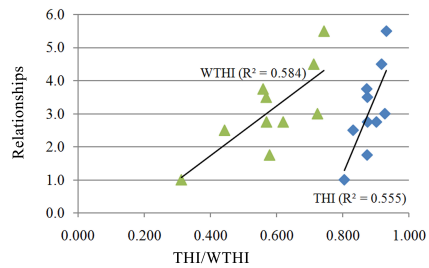
d) Impact of THI and WTHI on correctness (professionals)



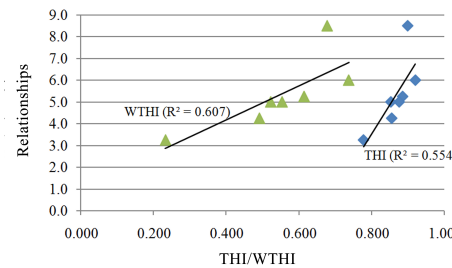
e) Impact of THI and WTHI on layout (students)



f) Impact of THI and WTHI on layout (professionals)



g) Impact of THI and WTHI on relationship (students)



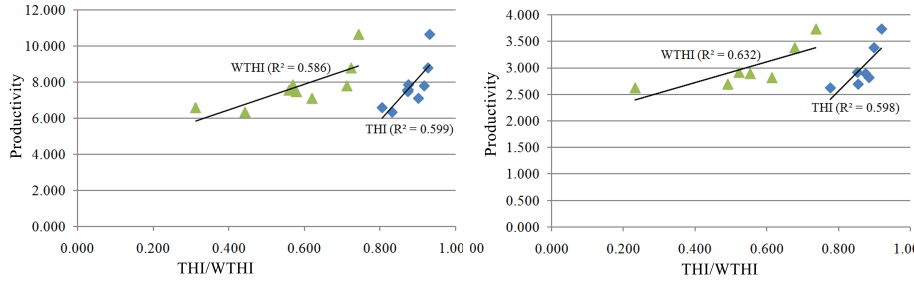
h) Impact of THI and WTHI on relationship (professionals)

Figure 6. Impact of THI and WTHI on analysis and design teams

Sr.	Teams	THI	WTHI	Und	Cor	Lay	AUR	Prod
1	Stu Team 1	0.874	0.579	6.6	5.6	6.4	1.8	7.487
2	Stu Team 2	0.932	0.743	7.4	6.6	8.4	5.5	10.667
3	Stu Team 3	0.927	0.723	7.2	6.6	7.0	3.0	8.800
4	Stu Team 4	0.902	0.620	6.0	5.8	6.8	2.8	7.111
5	Stu Team 5	0.875	0.569	7.0	6.2	5.6	3.5	7.595
6	Stu Team 6	0.917	0.712	8.2	7.6	8.4	4.5	7.805
7	Stu Team 7	0.805	0.312	5.0	4.8	5.4	1.0	6.593
8	Stu Team 8	0.832	0.443	6.2	5.4	6.8	2.5	6.344
9	Stu Team 9	0.875	0.570	6.8	6.8	6.4	2.8	7.865
10	Stu Team 10	0.873	0.559	6.2	6.0	5.8	3.8	7.564
11	Pro Team 1	0.885	0.614	5.8	6.2	6.4	5.3	2.818
12	Pro Team 2	0.899	0.678	7.2	8.0	7.6	8.5	3.381
13	Pro Team 3	0.777	0.232	4.4	5.2	4.8	3.3	2.625
14	Pro Team 4	0.920	0.737	5.8	7.6	6.4	6.0	3.733
15	Pro Team 5	0.855	0.491	6.4	5.0	5.8	4.3	2.695
16	Pro Team 6	0.876	0.553	6.0	6.2	6.0	5.0	2.895
17	Pro Team 7	0.853	0.522	5.4	5.4	5.6	5.0	2.917
AUR = Appropriately Used Relationships, Cor = Correctness, Lay = Layout, Prod = Productivity, Pro = Professionals, Stu = Students, Und = Understandability								

Table 3. Results of analysis and design phase

positive relationship with productivity. This indicates the contribution of weights assigned and supports the hypothesis HE1 for professionals. In the case of students, hypothesis HE0 cannot be rejected.



a) Impact of THI and WTHI on productivity (students)

b) Impact of THI and WTHI on productivity (professionals)

Figure 7. Impact of THI and WTHI on analysis and design teams' productivity

6.2 Implementation Phase

The results of the implementation phase also appear favorable. Table 4 provides the data related to the quality of implemented projects and productivity of teams. Weighted sum of bugs (WSB) is provided in column 5 and Defect Density is given in column 6. Maintainability Index (MI) and Cyclomatic Complexity are provided in the last two columns. The last two columns contain the values for productivity and FP productivity of implementation teams.

Sr.	Teams	THI	WTHI	WSB	DD	MI	CC	P	FP
1	Stu Team 1	0.932	0.743	48	0.009	73.330	2.3	0.47	0.96
2	Stu Team 2	0.863	0.501	98	0.023	78.235	1.7	0.37	0.68
3	Stu Team 3	0.906	0.673	64	0.013	61.235	1.5	0.35	0.79
4	Stu Team 4	0.849	0.464	99	0.025	53.255	2.6	0.30	0.58
5	Stu Team 5	0.893	0.620	89	0.028	63.425	1.7	0.32	0.48
6	Stu Team 6	0.891	0.613	70	0.020	67.036	2.5	0.30	0.50
7	Stu Team 7	0.872	0.552	92	0.041	45.651	1.6	0.32	0.30
8	Stu Team 8	0.908	0.675	58	0.016	43.392	2.8	0.40	0.52
9	Stu Team 9	0.833	0.426	140	0.057	60.424	3.9	0.25	0.35
10	Stu Team 10	0.889	0.609	97	0.043	47.235	2.3	0.38	0.34
11	Stu Team 11	0.863	0.508	93	0.043	64.936	2.6	0.36	0.36
12	Stu Team 12	0.895	0.649	91	0.026	55.456	2.2	0.37	0.67
13	Stu Team 13	0.900	0.696	85	0.026	76.253	1.9	0.33	0.46
14	Stu Team 14	0.866	0.511	91	0.026	61.436	1.6	0.33	0.50
15	Stu Team 15	0.919	0.655	87	0.023	77.219	1.5	0.29	0.56
16	Stu Team 16	0.847	0.440	114	0.075	57.945	2.5	0.26	0.24
17	Stu Team 17	0.932	0.782	91	0.016	79.548	2.3	0.41	0.85
18	Stu Team 18	0.785	0.232	146	0.062	52.235	4.5	0.21	0.337
19	Pro Team 1	0.885	0.614	131	0.016	72.050	4.8	0.31	0.95
20	Pro Team 2	0.899	0.678	123	0.013	78.820	4	0.43	0.95
21	Pro Team 3	0.777	0.232	156	0.021	66.040	5.6	0.28	0.74
22	Pro Team 4	0.920	0.737	106	0.012	85.670	2.6	0.42	0.95
23	Pro Team 5	0.855	0.491	173	0.020	77.890	4.9	0.28	0.92
24	Pro Team 6	0.876	0.553	133	0.016	69.390	4.7	0.29	0.90
25	Pro Team 7	0.853	0.522	138	0.016	76.650	4.9	0.29	0.91

CC = Cyclomatic Complexity, DD = Defect Density, Pro = Professionals,
MI = Maintainability Index, Stu = Students, WSB = Weighted Sum of Bugs,
P = Productivity, FP = Function Point Productivity

Table 4. Results of implementation phase

Figure 8 shows the impact of THI and WTHI on the quality of implemented projects. Figures 8 a), 8 b), 8 c) and 8 d) depict the impact on the weighted sum of bugs and defect density of implemented projects. The descending slopes of trend-lines clearly support HB1 and HD1 and indicate that the teams with higher THI

and WTHI values developed software with lower weighted bugs and defect density. It is also evident from these figures that HF1 is supported for professionals only.

The effects of THI and WTHI on maintainability index and cyclomatic complexity are shown in Figures 8 e), 8 f), 8 g) and 8 h). The upward sloping trend-lines for maintainability index and downward slopping trend-lines for cyclomatic complexity indicate that the teams with greater THI and WTHI values developed projects with greater maintainability and lower complexity. Hence, these figures support our HB1 and HD1 hypotheses. HF1, again, is supported for professionals only.

Figure 9 shows scatter plots with trend-lines that depict the relationship of THI and WTHI with productivity and FP productivity. These upward sloping trend-lines for productivity and FP productivity indicate that teams with higher values of THI and WTHI are more productive. These findings support our HA1 and HC1 hypotheses. In the case of professionals' teams only, R^2 values for WTHI are greater than those for THI for both FP productivity and productivity. Thus, hypothesis HE1 is supported for professionals only.

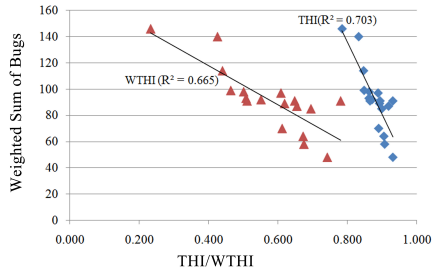
6.3 Testing Phase

The details regarding the testing are presented in Table 5. Figure 10 shows the impact of THI and WTHI on quality of testing. It is clear from these upward sloping trend-lines in Figures 10 a) and 10 b) that THI and WTHI have a positive correlation with architectural coverage. Hence, hypotheses HB1 and HD1 are supported for both students' and professionals' teams. HF1 is supported for professionals' teams only.

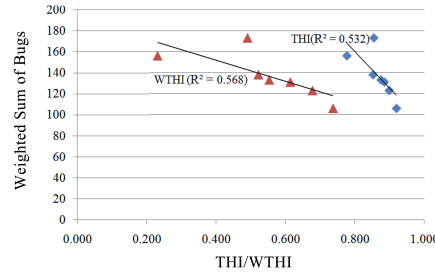
Figures 10 c), 10 d), 10 e) and 10 f) show scatter plots with associated trend-lines that display the impact of THI and WTHI on total number of defects uncovered by testing teams and conformity to provided test case template. Clearly, teams with higher values of THI and WTHI uncovered more defects and followed the given test case template more strictly. Hence, HB1 and HD1 are supported. HF1 is also supported for both types of teams.

Figure 11 shows the relationships between THI and productivity and WTHI and productivity of testing teams. It is clear from this figure that the teams with greater THI and WTHI values appear more productive. These values support our HA1 and HC1 hypotheses. In the case of professionals' teams only, the R^2 values for WTHI are greater than those for THI. This shows that hypothesis HE1 is supported for professionals' teams only.

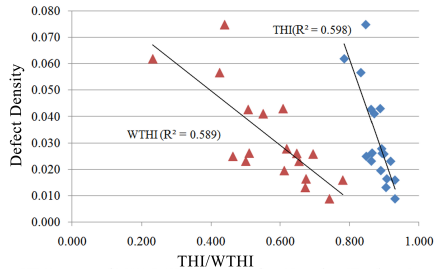
Our research focuses on the impact of team homogeneity (i.e. WTHI) on team productivity and software quality. Our results imply that human factors (i.e. personality aspects or team homogeneity) should be taken into consideration while assigning jobs to existing employees or hiring new personnel. WTHI can help the software industry managers during the team composition process. Our results indicate that teams with higher values of THI and WTHI are more productive and produced better quality software. Our results also reveal that WTHI is more strongly correlated with software quality and team productivity for professionals' teams in comparison with THI. The results of the industry survey conducted to determine



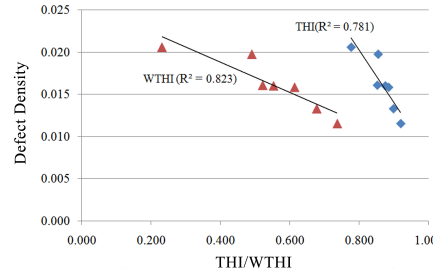
a) Impact of THI and WTHI on weighted sum of bugs (students)



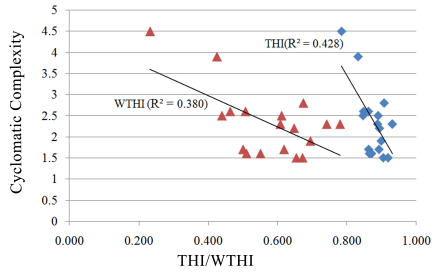
b) Impact of THI and WTHI on weighted sum of bugs (professionals)



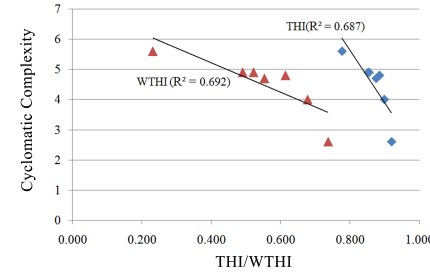
c) Impact of THI and WTHI on defect density (students)



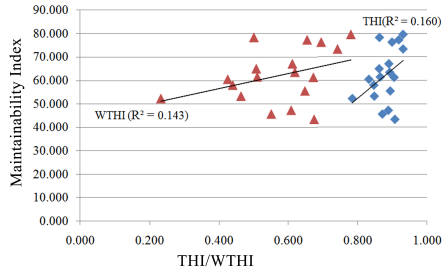
d) Impact of THI and WTHI on defect density (professionals)



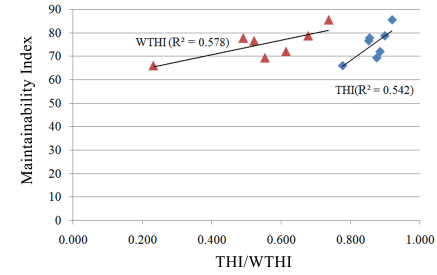
e) Impact of THI and WTHI on cyclomatic complexity (students)



f) Impact of THI and WTHI on cyclomatic complexity (professionals)

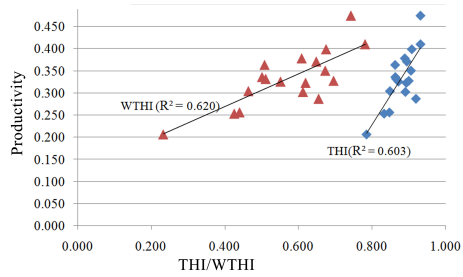


g) Impact of THI and WTHI on maintainability index (students)

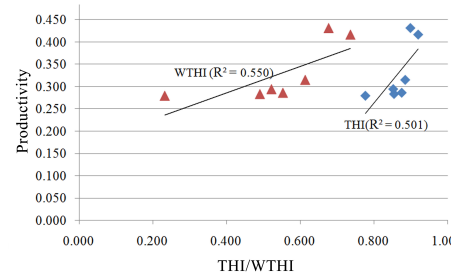


h) Impact of THI and WTHI on maintainability index (professionals)

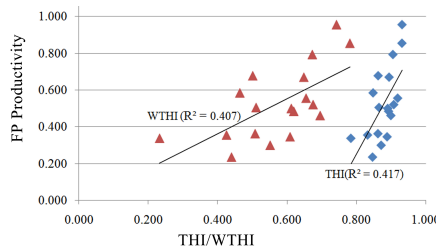
Figure 8. Impact of THI and WTHI on implemented projects' quality



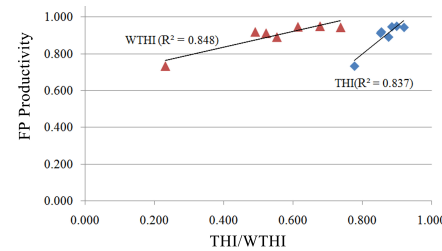
a) Impact of THI and WTHI on productivity (students)



b) Impact of THI and WTHI on productivity (professionals)



c) Impact of THI and WTHI on FP productivity (students)



d) Impact of THI and WTHI on FP productivity (professionals)

Figure 9. Impact of THI and WTHI on productivity of implementation teams

the weights of personality traits show that "Openness to Experience", "Agreeableness", and "Conscientiousness" are the top three most important traits whereas the "Neuroticism" is the least important trait.

6.4 Hypotheses Testing

We have investigated our null hypotheses based on one-way analysis of variance (ANOVA) test [18] to analyze the significant difference between independent and dependent variables for students' and professionals' teams. THI and WTHI are our independent variables whereas different components of team productivity and software quality are our dependent variables. A threshold p-value of 0.05 was used.

Table 6 provides the details of the 60 hypothesis tests (30 for THI and 30 for WTHI) we have conducted using the IBM SPSS tool [44] for different factors of software quality and team productivity. It can be seen that in the case of productivity during the analysis and design phase, the null hypothesis (HA0) can be rejected for both students' and professionals' teams. For almost all of the quality factors considered for analysis and design models, hypotheses (HB1 and HD1) can be accepted. The only exception is understandability (for both THI and WTHI in case of professionals). Similarly, for team productivity during the implementation

Sr.	Teams	THI	WTHI	AC	DU	TCC	Prod
1	Stu Team 1	0.808	0.328	37.255	5	83.239	1.508
2	Stu Team 2	0.850	0.444	50.980	6	84.244	1.898
3	Stu Team 3	0.905	0.680	60.784	8	91.364	2.952
4	Stu Team 4	0.873	0.577	35.294	6	86.898	1.552
5	Stu Team 5	0.827	0.386	27.451	4	85.795	0.950
6	Stu Team 6	0.883	0.585	50.980	4	90.385	2.122
7	Stu Team 7	0.799	0.312	19.608	2	80.519	0.654
8	Stu Team 8	0.825	0.396	29.412	5	87.190	1.136
9	Stu Team 9	0.896	0.664	47.059	7	92.375	1.691
10	Stu Team 10	0.807	0.337	23.529	3	88.462	1.371
11	Stu Team 11	0.903	0.650	23.529	5	89.744	2.200
12	Stu Team 12	0.888	0.611	39.216	6	87.500	2.139
13	Stu Team 13	0.862	0.507	33.333	5	90.476	2.061
14	Stu Team 14	0.856	0.478	33.333	4	84.921	1.789
15	Stu Team 15	0.878	0.563	37.255	5	90.705	1.818
16	Pro Team 1	0.885	0.614	111.765	12	91.390	1.606
17	Pro Team 2	0.899	0.678	131.373	16	93.006	1.775
18	Pro Team 3	0.777	0.232	70.588	11	88.060	1.059
19	Pro Team 4	0.920	0.737	101.961	17	94.921	2.306
20	Pro Team 5	0.855	0.491	82.353	11	87.619	1.331
21	Pro Team 6	0.876	0.553	111.765	12	93.704	1.755
22	Pro Team 7	0.853	0.522	100.000	3	91.870	1.709
AC = Architectural Coverage , DU = Defects Uncovered , Pro = Professionals, Prod = Productivity, TCC = Test Case Conformity, Stu = Students							

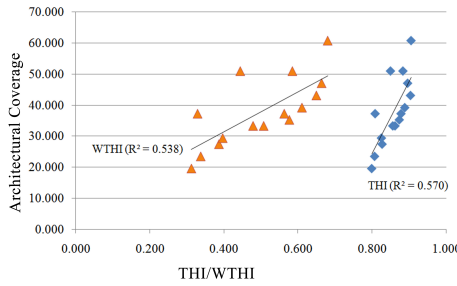
Table 5. Results of testing phase

phase null hypothesis (HA0) is rejected for productivity and FP productivity in all cases except for the productivity of professionals' teams (for THI).

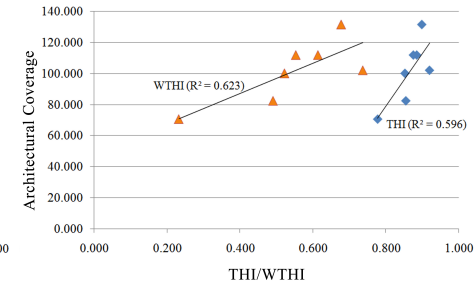
In case of quality factors during the implementation phase, we cannot reject the null hypothesis HB0 only for the weighted sum of bugs (for THI in case of professionals) and maintainability index (for both THI and WTHI). For the testing phase, the results are more promising for both productivity and quality. The null hypothesis cannot be rejected only for THI in the case of professionals' teams' defects uncovered. Out of 60 tests, we are unable to reject the null hypotheses for just 8 cases. Furthermore, in 5 out of those 8 cases overall, null hypotheses are rejected for only professionals' teams for THI. This shows the importance of using weights especially for professionals' teams.

7 THREATS TO VALIDITY

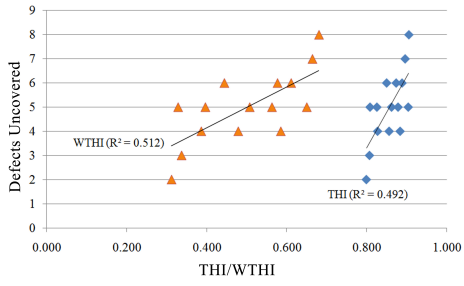
Despite the fact that our results seem promising, some factors may threaten their validity. First of all, in the case of students, subjects' competence, intelligence,



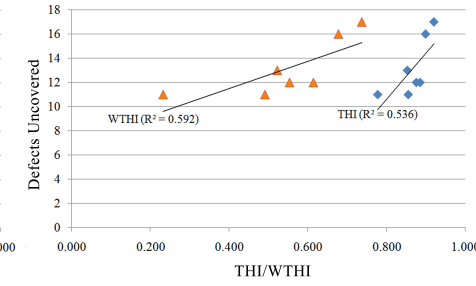
a) Impact of THI and WTHI on architectural coverage (students)



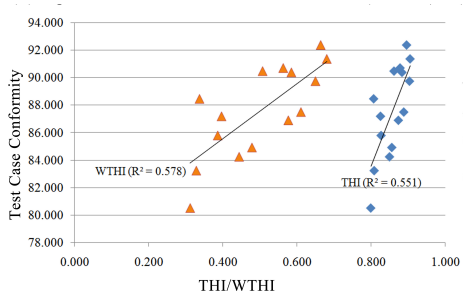
b) Impact of THI and WTHI on architectural coverage (professionals)



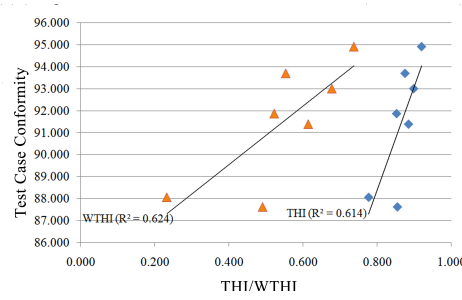
c) Impact of THI and WTHI on defects uncovered (students)



d) Impact of THI and WTHI on defects uncovered (professionals)



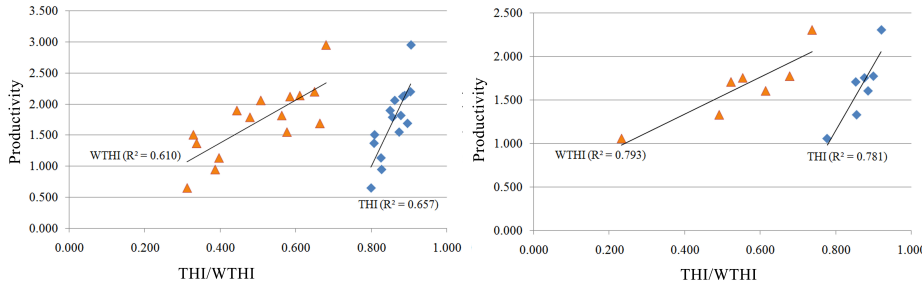
e) Impact of THI and WTHI on test case (students)



f) Impact of THI and WTHI on test case (professionals)

Figure 10. Impact of THI and WTHI on testing teams' test cases' quality

learning ability, degree of friendship, previous domain knowledge, programming experience, interest in programming, and gender can influence team's productivity and quality of projects. To avoid these threats, we formulated teams by making three buckets of CGPA (I. 3 or above, II. 2.50 to 3.00, III. 2.00 to 2.50) and randomly selected one member from bucket I and two members each from buckets II and III. At most one member with some previous experience was part of a single team. Also, there was no more than one female member in each team. Moreover,



a) Impact of THI and WTHI on productivity (students)

b) Impact of THI and WTHI on productivity (professionals)

Figure 11. Impact of THI and WTHI on productivity of testing teams

it was made sure that the average CGPA of teams was within a specific range (i.e. 2.50 to 2.80).

In the case of professionals, level of experience, academic qualifications, and domain exposure can affect team productivity and project quality. These threats were mitigated by making sure that every team member has a Bachelors degree in Computer Science. Furthermore, the average experience of a team was kept between 3 to 4 years and no more than one female member was assigned to a team.

Last, but not the least, the selection of different projects for each iteration of a specific phase of SDLC could have made our results incomparable. We avoided this threat by selecting the same project for all iterations of the experiment for each phase of SDLC. Moreover, using the same project for professionals as well as students could have compromised team productivity and project quality. We eliminated this threat by selecting a relatively more complex project for professionals. This helped us in avoiding threats to external validity thereby making our results generalizable to both academic and industrial environments.

8 CONCLUSIONS AND FUTURE WORK

In this research we introduced a new metric called WTHI (derived from THI). A survey was conducted in the Pakistani software industry and weights were determined for all five personality factors. The impact of THI and WTHI was compared for analysis and design, implementation, and testing phases of SDLC in academic and industrial environments.

The results of this study reveal that teams with greater values of THI and WTHI performed better in almost all the phases of SDLC. A positive correlation of team productivity and software quality was observed with both THI and WTHI. These results also indicate that, as compared to THI, WTHI is more strongly correlated with team productivity and software quality for all the productivity and quality factors in the case of teams comprising professionals.

Sr.	Dependent Variables	Sub	F	THI (Sig)	F	WTHI (Sig)
Analysis and Design Phase						
1	Productivity	Stu	11.99	0.01	11.34	0.01
2		Pro	7.45	0.04	8.62	0.03
3	Understandability	Stu	13.45	0.01	20.43	0.00
4		Pro	5.37	0.07	5.11	0.07
5	Correctness	Stu	12.79	0.01	15.90	0.00
6		Pro	7.11	0.04	7.96	0.04
7	Layout	Stu	8.83	0.02	9.69	0.01
8		Pro	10.58	0.02	11.92	0.02
9	Relationships	Stu	10.00	0.01	11.25	0.01
10		Pro	5.53	0.05	6.83	0.04
Implementation Phase						
11	Productivity	Stu	24.33	0.00	26.21	0.00
12		Pro	5.02	0.08	6.12	0.05
13	FP Productivity	Stu	11.48	0.00	11.01	0.00
14		Pro	25.69	0.00	28.05	0.00
15	Weighted Sum of Bugs	Stu	38.01	0.00	31.79	0.00
16		Pro	5.68	0.06	6.58	0.05
17	Defect Density	Stu	23.80	0.00	23.00	0.00
18		Pro	17.86	0.01	23.36	0.00
19	Cyclomatic Complexity	Stu	11.97	0.00	9.84	0.01
20		Pro	11.02	0.02	11.27	0.02
21	Maintainability Index	Stu	3.06	0.10	2.68	0.12
22		Pro	5.92	0.06	6.68	0.05
Testing Phase						
23	Productivity	Stu	24.92	0.00	20.39	0.00
24		Pro	17.87	0.01	19.23	0.01
25	Architectural Coverage	Stu	17.25	0.00	15.18	0.00
26		Pro	7.39	0.04	8.27	0.03
27	Defects Uncovered	Stu	12.61	0.00	13.65	0.00
28		Pro	5.79	0.06	7.26	0.04
29	Test Case confirmity	Stu	15.98	0.00	17.82	0.00
30		Pro	7.98	0.04	8.32	0.03
Sig. = Significance of F or p-value, Sub = Subject, Pro = Pro, Stu = Students						

Table 6. Results of ANOVA tests

Future work in this direction may focus on evaluating the impact of WTHI on other phases of SDLC such as requirements engineering and maintenance. It would also be interesting to replicate this experiment using more complex projects and larger team sizes (i.e. more than five members in a team). Last, but not the least, other personality models (e.g. MBTI, KTS, etc.) may also be used to determine team homogeneity.

REFERENCES

[1] Personality Test Based on Five Factor Model.

[2] A Study on Personality Traits' Rating.

[3] Time Keeper.

[4] Acuña, S. T.—Gómez, M.—Juristo, N.: How Do Personality, Team Processes and Task Characteristics Relate to Job Satisfaction and Software Quality? Information and Software Technology, Vol. 51, 2009, No. 3, pp. 627–639, doi: 10.1016/j.infsof.2008.08.006.

[5] Acuña, S. T.—Gómez, M. N.—Hannay, J. E.—Juristo, N.—Pfahl, D.: Are Team Personality and Climate Related to Satisfaction and Software Quality? Aggregating Results from a Twice Replicated Experiment. Information and Software Technology, Vol. 57, 2015, pp. 141–156, doi: 10.1016/j.infsof.2014.09.002.

[6] Acuña, S. T.—Juristo, N.: Assigning People to Roles in Software Projects. Software: Practice and Experience, Vol. 34, 2004, No. 7, pp. 675–696, doi: 10.1002/spe.586.

[7] Barroso, A. S.—Madureira, J. S.—Soares, M. S.—do Nascimento, R. P. C.: Influence of Human Personality in Software Engineering – A Systematic Literature Review. Proceedings of the 19th International Conference on Enterprise Information Systems (ICEIS), 2017, Vol. 1, pp. 53–62, doi: 10.5220/0006292000530062.

[8] Bell, S. T.: Deep-Level Composition Variables as Predictors of Team Performance: A Meta-Analysis. Journal of Applied Psychology, Vol. 92, 2007, No. 3, pp. 595–612, doi: 10.1037/0021-9010.92.3.595.

[9] Bradley, J. H.—Hebert, F. J.: The Effect of Personality Type on Team Performance. Journal of Management Development, Vol. 16, 1997, No. 5, pp. 337–353, doi: 10.1108/02621719710174525.

[10] Capretz, L. F.: Personality Types in Software Engineering. International Journal of Human-Computer Studies, Vol. 58, 2003, No. 2, pp. 207–214, doi: 10.1016/s1071-5819(02)00137-4.

[11] Capretz, L. F.—Ahmed, F.: Making Sense of Software Development and Personality Types. IT Professional, Vol. 12, 2010, No. 1, pp. 6–13, doi: 10.1109/mitp.2010.33.

[12] Choi, K. S.—Deek, F. P.—Im, I.: Exploring the Underlying Aspects of Pair Programming: The Impact of Personality. Information and Software Technology, Vol. 50, 2000, No. 11, pp. 1114–1126, doi: 10.1016/j.infsof.2007.11.002.

[13] Cruz, S.—da Silva, F. Q. B.—Capretz, L. F.: Forty Years of Research on Personality in Software Engineering: A Mapping Study. Computers in Human Behavior, Vol. 46, 2015, pp. 94–113, doi: 10.1016/j.chb.2014.12.008.

[14] DeMarco, T.—Lister, T.: Peopleware: Productive Projects and Teams. Third Edition. Addison-Wesley, 2013.

[15] Galin, D.: Software Quality Assurance: From Theory to Implementation. Pearson Education India, 2004.

[16] Goldberg, L. R.—Johnson, J. A.—Eber, H. W.—Hogan, R.—Ashton, M. C.—Cloninger, C. R.—Gough, H. G.: The International Personality Item Pool and the Future of Public-Domain Personality Mea-

sures. Journal of Research in Personality, Vol. 40, 2006, No. 1, pp. 84–96, doi: 10.1016/j.jrp.2005.08.007.

[17] Gorla, N.—Lam, Y. W.: Who Should Work with Whom? Building Effective Software Project Teams. Communications of the ACM, Vol. 47, 2004, No. 6, pp. 79–82, doi: 10.1145/990680.990684.

[18] James, G.—Witten, D.—Hastie, T.—Tibshirani, R.: An Introduction to Statistical Learning. Springer Texts in Statistics Book Series, Vol. 103, 2013, doi: 10.1007/978-1-4614-7138-7.

[19] Karasneh, B.—Stikkolorum, D.—Larios, E.—Chaudron, M.: Quality Assessment of UML Class Diagrams. Proceedings of the MODELS Educators Symposium 2015, Ottawa, Canada, 2015. CEUR Workshop Proceedings, Vol. 1555, 2015.

[20] Karn, J.—Cowling, T.: A Follow Up Study of the Effect of Personality on the Performance of Software Engineering Teams. Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering (ISESE '06), 2006, pp. 232–241, doi: 10.1145/1159733.1159769.

[21] Karn, J. S.—Syed-Abdullah, S.—Cowling, A. J.—Holcombe, M.: A Study Into the Effects of Personality Type and Methodology on Cohesion in Software Engineering Teams. Behaviour and Information Technology, Vol. 26, 2007, No. 2, pp. 99–111, doi: 10.1080/01449290500102110.

[22] Keirsey, D.—Bates, M.: Please Understand Me: Character and Temperament Types. Prometheus Nemesis Books, Del Mar, CA, 1978.

[23] Kichuk, S. L.—Wiesner, W. H.: The Big Five Personality Factors and Team Performance: Implications for Selecting Successful Product Design Teams. Journal of Engineering and Technology Management, Vol. 14, 1997, No. 3-4, pp. 195–221, doi: 10.1016/s0923-4748(97)00010-6.

[24] Lenberg, P.—Feldt, R.—Wallgren, L. G.: Behavioral Software Engineering: A Definition and Systematic Literature Review. Journal of Systems and Software, Vol. 107, 2015, pp. 15–37, doi: 10.1016/j.jss.2015.04.084.

[25] Levine, J. M.—Moreland, R. L.: Progress in Small Group Research. Annual Review of Psychology, Vol. 41, 1990, No. 1, pp. 585–634, doi: 10.1146/annurev.ps.41.020190.003101.

[26] McCrae, R. R.—John, O. P.: An Introduction to the Five-Factor Model and Its Applications. Journal of Personality, Vol. 60, 1992, No. 2, pp. 175–215, doi: 10.1111/j.1467-6494.1992.tb00970.x.

[27] Meyer, A. N.—Fritz, T.—Murphy, G. C.—Zimmermann, T.: Software Developers' Perceptions of Productivity. Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014), 2014, pp. 19–29, doi: 10.1145/2635868.2635892.

[28] Myers, I. B.—McCaulley, M. H.—Quenk, N. L.—Hammer, A. L.: MBTI Manual: A Guide to the Development and Use of the Myers-Briggs Type Indicator. Third Edition. Consulting Psychologists Press, Palo Alto, CA, 1998.

[29] Peeters, M. A.—Van Tuijl, H. F.—Rutte, C. G.—Reymen, I. M.: Personality and Team Performance: A Meta-Analysis. European Journal of Personality, Vol. 20, 2006, No. 5, pp. 377–396, doi: 10.1002/per.588.

[30] Pervin, L. A.—John, O. P.: Handbook of Personality: Theory and Research. Second Edition. Elsevier, 1999.

[31] Peslak, A. R.: The Impact of Personality on Information Technology Team Projects. Proceedings of the 2006 ACM SIGMIS CPR Conference on Computer Personnel Research: Forty Four Years of Computer Personnel Research: Achievements, Challenges and the Future (SIGMIS CPR '06), 2006, pp. 273–279, doi: 10.1145/1125170.1125233.

[32] Pieterse, V.—Kourie, D. G.—Sonnekus, I. P.: Software Engineering Team Diversity and Performance. Proceedings of the 2006 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries (SAICSIT '06), 2006, pp. 180–186, doi: 10.1145/1216262.1216282.

[33] Poonam, R.—Yasser, C. M.: An Experimental Study to Investigate Personality Traits on Pair Programming Efficiency in Extreme Programming. 2018 5th International Conference on Industrial Engineering and Applications (ICIEA), IEEE, 2018, pp. 95–99, doi: 10.1109/iea.2018.8387077.

[34] Pressman, R. S.: Software Engineering: A Practitioner's Approach. Sixth Edition. Palgrave Macmillan, 2005.

[35] Qamar, N.—Malik, A. A.: Birds of a Feather Gel Together: Impact of Team Homogeneity on Software Quality and Team Productivity. IEEE Access, Vol. 7, 2019, pp. 96827–96840, doi: 10.1109/access.2019.2929152.

[36] Qamar, N.—Malik, A. A.: Evaluating the Impact of Pair Testing on Team Productivity and Test Case Quality – A Controlled Experiment. Pakistan Journal of Engineering and Applied Sciences, Vol. 25, 2019, pp. 80–88.

[37] Richardson, I.—Casey, V.—McCaffery, F.—Burton, J.—Beecham, S.: A Process Framework for Global Software Engineering Teams. Information and Software Technology, Vol. 54, 2012, No. 11, pp. 1175–1191, doi: 10.1016/j.infsof.2012.05.002.

[38] Rutherfoord, R. H.: Using Personality Inventories to Help Form Teams for Software Engineering Class Projects. ACM SIGCSE Bulletin, Vol. 33, 2001, No. 3, pp. 73–76, doi: 10.1145/507758.377486.

[39] Salleh, N.—Mendes, E.—Grundy, J.: The Effects of Openness to Experience on Pair Programming in a Higher Education Context. 2011 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE & T), IEEE, 2011, pp. 149–158, doi: 10.1109/cseet.2011.5876082.

[40] Salleh, N.—Mendes, E.—Grundy, J.—Burch, G. S. J.: The Effects of Neuroticism on Pair Programming: An Empirical Study in the Higher Education Context. Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '10), 2010, Art. No. 22, 10 pp., doi: 10.1145/1852786.1852816.

[41] Salleh, N.—Mendes, E.—Grundy, J.—Burch, G. S. J.: An Empirical Study of the Effects of Conscientiousness in Pair Programming Using the Five-Factor Personality Model. Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE '10), 2010, Vol. 1, pp. 577–586, doi: 10.1145/1806799.1806883.

[42] Sfetsos, P.—Stamelos, I.—Angelis, L.—Deligiannis, I.: An Experimental Investigation of Personality Types Impact on Pair Effectiveness in Pair Programming. Empirical Software Engineering, Vol. 14, 2009, No. 2, Art. No. 187, doi: 10.1007/s10664-008-9093-5.

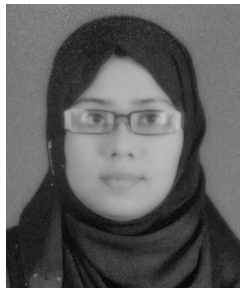
[43] Sudhakar, G. P.—Farooq, A.—Patnaik, S.: Measuring Productivity of Software Development Teams. Serbian Journal of Management, Vol. 7, 2012, No. 1, pp. 65–75, doi: 10.5937/sjm1201065s.

[44] IBM SPSS Statistics.

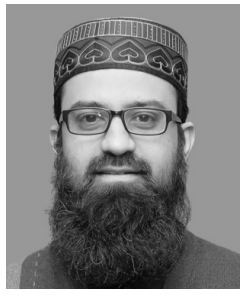
[45] Walle, T.—Hannay, J. E.: Personality and the Nature of Collaboration in Pair Programming. 2009 3rd International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, FL, USA, IEEE, 2009, pp. 203–213, doi: 10.1109/esem.2009.5315996.

[46] Yilmaz, M.—O'Connor, R. V.—Clarke, P.: Effective Social Productivity Measurements During Software Development – An Empirical Study. International Journal of Software Engineering and Knowledge Engineering, Vol. 26, 2016, No. 3, pp. 457–490, doi: 10.1142/s0218194016500194.

[47] Yilmaz, M.—O'Connor, R. V.—Colomo-Palacios, R.—Clarke, P.: An Examination of Personality Traits and How They Impact on Software Development Teams. Information and Software Technology, Vol. 86, 2017, pp. 101–122, doi: 10.1016/j.infsof.2017.01.005.



Nosheen Qamar is currently working as Assistant Professor at the Department of Computer Science and Information Technology, the University of Lahore, Lahore, Pakistan. She received her Ph.D. degree in computer science from the National University of Computer and Emerging Sciences (FAST-NUCES), Lahore, Pakistan in 2020. Before joining the University of Lahore, she has gained 7 years of industry experience with last job title as Software Development Manager. Her research interests include software engineering, design patterns, software teams, project management, empirical software engineering, and requirements engineering.



Ali Afzal Malik is currently working as Assistant Professor and Head of the Computer Science Department of the National University of Computer and Emerging Sciences (FAST-NUCES). He started his professional career in 2003 working as Software Engineer in Techlogix – a well-reputed Pakistani software house. After receiving the prestigious Fulbright scholarship in 2005, he obtained M.S. and Ph.D. degrees in computer science from the University of Southern California (USC), Los Angeles, USA in 2007 and 2010, respectively. He was awarded the Office of International Services (OIS) Academic Achievement Award twice (2007 and 2010) during his stay at USC. His research paper on the quantitative aspects of requirements elaboration was given the best paper award in SBES 2008 (Sao Paulo, Brazil). Before joining FAST-NUCES in 2013, he has held an adjunct faculty position at the Lahore University of Management Sciences (LUMS) and a full-time faculty position at the University of Central Punjab (UCP). He has undertaken research in software cost estimation at two of the world's leading research centers in software engineering i.e. USC's Center for Systems and Software Engineering (CSSE) and Institute of Software, Chinese Academy of Sciences (ISCAS). His current research work focuses on areas such as empirical software engineering, requirements engineering, and software cost estimation. He is a senior member of IEEE.

Computing and Informatics, Vol. 39, 2020, 1022–1060, doi: 10.31577/cai_2020_5_1022

FORMAL VERIFICATION OF UML MARTE SPECIFICATIONS BASED ON A TRUE CONCURRENCY REAL TIME MODEL

Nadia Chabbat

LAAS Laboratory, Department of Computer Science
University of Badji Mokhtar, Annaba, Algeria
e-mail: lydia.chab@hotmail.fr

Djamel Eddine Saidouni, Radja Boukharrou

MISC Laboratory, Department of Computer Science
University of Abdelhamid Mehri, Constantine 2, Algeria
e-mail: {djamel.saidouni, radja.boukharrou}@univ-constantine2.dz

Salim Ghanemi

LAAS Laboratory, Department of Computer Science
University of Badji Mokhtar
Annaba, Algeria
e-mail: ghanemisalim@univ-annaba.dz

Abstract. For critical embedded systems the formal validation and verification is required. However, the real-time model checking suffers from problems of state-space explosion and clock explosion. The aim of this paper is to ensure an improvement of the Modeling and Analysis of Real-Time Embedded systems (MARTE), which is de facto standard, with formal semantics for verification finality. Therefore, we propose an operational method for translating UML sequence diagrams with MARTE annotations to Time Petri nets with Action Duration specifications (DTPN). Based on true concurrency semantics, the semantics of these specifications are defined in terms of Duration Action Timed Automata (daTA).

Keywords: Real-time embedded system, UML MARTE, DTPN, duration action timed automata, parallel computing, sequence diagram, formal verification

1 INTRODUCTION

Real-time embedded systems solve behaviors constrained by time. These systems provide a specific function in a much larger system within time consideration [1, 2]. The design of such systems is associated with time constraints specification. Real-time embedded systems are often critical and require the modeling of real-time response at the functional level. The assessment of such systems may be achieved through verification and validation processes based on robust formal approaches to meet the required timed constrained functional system properties.

Several studies have proposed a model-based engineering approaches that offer advanced modeling mechanisms such as UML (Unified Modeling Language) [3]. MARTE is an OMG UML profile dedicated for modeling and analysis of real-time embedded systems [4, 5]. MARTE specifies concepts for characterizing UML elements in order to model software and hardware platforms, resources, and quantitative characteristics such as execution time. However, once the software is modeled, the difficulty lies in the expression of appropriate properties and formal checks. In order to meet these requirements, formal analytical approaches have been developed so to integrate or extend formal models in the designing process of real-time embedded systems developed with MARTE. Both works presented in [6, 7] use the Time Petri Net analyzer [8] (TINA) model-checking tool to verify temporal properties on structural and behavioral diagrams specified in MARTE.

Some work, as [9, 10], extended a formal language in order to take into consideration time aspects. In [9], the discrete-time of Promela language has been extended by a variable, named *"timer"*, that corresponds to the discrete-time *"countdown"*. However, the extended model is difficult to use for representing the coincidence clock tickings. Another work, presented in [10], proposes an extension of Promela language with discrete-time to allow the verification by SPIN model checker [11].

In [12], an interesting approach has been proposed. It interprets parallel activities, modeled in MARTE sequence diagrams, by parallel transitions in a Petri net like specification. More precisely, the specification is written in timed colored Petri nets with inhibitor arcs model (TCPNIA) [13]. In the later work, the duration of an activity is integrated as a constraint interval associated with the corresponding transition of the Petri net. To verify some properties, TCPNIA specification is translated to a Timed Automata [14, 15] in order to use some existing model checker tools as SMV [16]. However, this approach only expresses activity duration without considering *latency*, *delay* and *congestion* time specification. Since Timed Automata is based on interleaving semantics, there is no way to express the parallel execution of two activities. To overcome such limitation, it is possible to interpret each activity having a non-null duration by two sequential transitions modeling the

activity's start and end events. Though this solution is correct, it presents some serious inconvenience. In fact, larger number of transitions in the Petri net specification leads to a significant clock number increase in the associated timed automata. Indeed, the number of zones, respectively regions, in the zone graph, respectively region graph, is exponential to the number of clocks [15]. Thus, this leads to the state space combinatorial explosion problem [17, 18, 19].

In our proposed approach, Time Petri Nets with action Duration (DTPN) specification model is used as a semantics model of MARTE SD specification [20]. In fact, DTPN has a true concurrency semantics and considers both timing constraints and duration of actions. The true concurrency semantics allows the consideration of activities duration without using the split technique. The underlying semantics model is a Durational Action Timed Automaton (daTA) [21].

This paper defines an operational method for translating MARTE SD specifications to DTPNs ones. As a consequence, on one hand, the transformation leads to a small DTPN specification and on the other hand, the verification of a DTPN specification is based on its daTA corresponding model, which allows the reuse of clocks. So, this lead to reduce, in some case avoid the problem of the combinatorial explosion of the number of graph states, which is one of the main limitations of applying model-checking methods on industrial-sized models and, to reduce of the verification time, that is often exponential for large-size systems.

The rest of this paper is structured as follows: In Section 2, we present preliminary definitions of MARTE sequence diagrams, Time Petri Nets with action duration and durational action timed automata. Section 3 defines the formal semantics of MARTE sequence diagrams and formal translation rules of basic elements and some combined fragments. In Section 4, we present a case study to better illustrate the interest of the proposed approach, as well as its implementation in practice. Section 5 discusses the advantages of the proposed approach with related works. Finally, Section 6 gives some conclusions and perspectives of this work.

2 PRELIMINARY DEFINITIONS

2.1 MARTE Sequence Diagrams

The UML sequence diagram is a form of interaction diagram. It allows the description of a specific interaction in terms of participating objects and sequence of messages exchanged along progress to perform desired activity. Graphically, an interaction is composed by two lifelines and a message. A lifeline represents an instance corresponding to a particular object that participates in the interaction. The events along a lifeline are, in general, partially ordered, the order in which these events will occur. A message represents a communication that transmits information between objects or an object and its environment. A message specifies the kind of a communication between objects as synchronous or asynchronous and notes the occurrences of the sending event at the sender and the receiving event at the receiver level. In this work, we will focus on the order and type of messages,

synchronous, asynchronous and reply. Additionally, we will take into consideration the timing constraints imposed on transmitted messages between objects involved in the interaction.

A sequence diagram describes only a fragment of the system behavior. The complete behavior of the system can be expressed by a set of sequence diagrams to specify all possible interactions. An interaction is a behavior unit that focuses on the observable transmissions of information between connected objects over time. Each interaction may be caused by actions executed by communicated objects. As defined in [22]: *an action takes a set of inputs and converts them into a set of outputs, though either or both sets may be empty.* For examples, an action can be a call operation, send signal, receive signal, write variable or read variable. The execution of actions can result by an event as a call or a signal event.

To get more complex interactions, combined fragments technique may be used. A combined fragment consists of an operator and a number of operands. Depending on the used operator, the number of operands is defined. For example, break, opt, loop, assert, ref and neg operators have one operand. Most other operators like alt, seq and par, have more than one operand. Fragments and their operators can be inductively combined for describing complex interactions.

In order to enrich UML models with annotations related to time (values and timing constraints), the UML MARTE profile provides basic and advanced time modeling concepts, such as stereotypes, which allow the consideration of temporal behaviors. This profile is intended to replace the existing UML SPT Profile (Schedulability, Performance and Time) [23], that is incompatible with UML 2 and MDA standards [24].

Figure 1 shows the different concepts in a sequence diagram used for specifying time and timing constraints in UML MARTE profile. These elements are defined in the SimpleTime package of CommonBehaviors. TimeObservation is a reference to an instant of time, while DurationObservation is a reference to a duration of an execution. TimeConstraints can be in the form of duration, as well as some instants of time; sequence diagram supports both. DurationConstraint defines a Constraint that refers to a duration interval, which defines the range between two duration. A duration defines a value specification that specifies the temporal distance between two time instants. Timing constraints are expressed between a pair of braces. The annotations in color are not part of the model, they are used to specify model elements. More details about this system example can be found in [25].

2.2 Time Petri Nets with Action Duration

Time Petri Nets with Action Duration can be considered as a generalization of Merlin's TPNs [27], T-TdPNs [28] and P-TdPN [29]. The basic idea of DTPN is to associate two date's *min* and *max* with each transition that define its firing interval (temporal interval). Although the firing of a transition is instantaneous, the execution duration of the action associated to this transition may have non-null duration. For example, let t be a transition associated to the action which

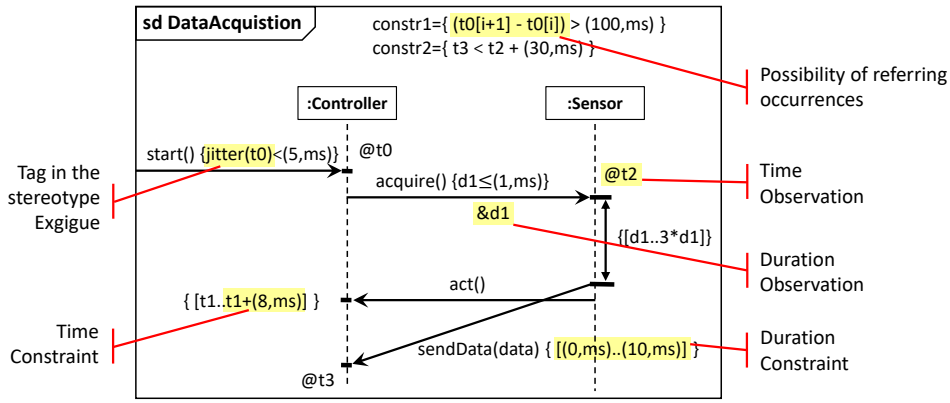
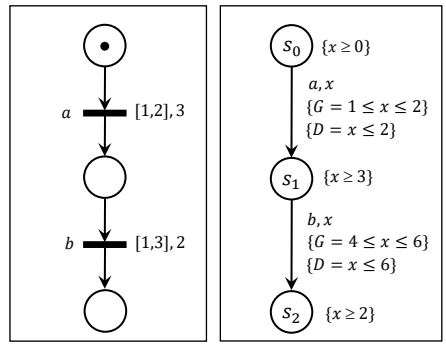


Figure 1. Time and timing constraints illustration [26]

has a duration d. If θ is the enabling date of t then the firing of t will be in the time interval $[\theta + min, \theta + max]$. The firing of t marks the start of execution of the associated action. Figure 2 a) shows an example of a time Petri nets with action duration.



a) DTPN b) daTA

Figure 2. DTPN and its corresponding daTA automata

A place of a DTPN corresponds to two sets: a set of *available tokens* or *free tokens* and a set of *unavailable tokens* or *bound tokens*. Unavailable tokens, put on the right side of a place, are bound to the firing of transitions associated to actions that are currently running. In a DTPN, an unavailable token becomes available if the end of execution of the action associated to the transition that produced this token is reached. A token in place p at the time ϑ becomes *available* (in the left side of p) at the time $\vartheta + d$. Thus, the token is bound to the firing of the transition during the interval $[\vartheta, \vartheta + d[$ and it becomes free at the time $\vartheta + d$.

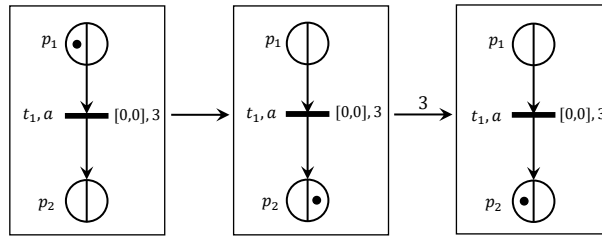


Figure 3. Marked DTPN

In Figure 3 a), the token in place P_1 is not bound to any transition. This token is called *free*. In the case when the transition would be fired, it could be argued that the action associated to the firing of t_1 has started its execution. This is marked by the presence of the token in place P_2 (Figure 3 b)). Thus, the token in place P_2 is bound to the firing of t_1, but after completion of the action a, i.e. after 3 units of time, this token will become free (Figure 3 c)). In a place, the set of free tokens will be denoted by FT, while bound tokens set will be denoted by BT.

Definition 1. Let $\mathbb{T}$ be a non-negative temporal domain, like $\mathbb{Q}^+$ or $\mathbb{R}^+$.

Definition 2. Let Act be a finite set of actions, i.e. an alphabet. A Time Perti Net with action Duration (DTPN) on $\mathbb{T}$ and of support Act is a tuple $\langle P, T, B, F, \lambda, SI, \Gamma \rangle$, such that:

- $Q = \langle P, T, B, F \rangle$ is Perti net, where P is a set of places, T is set of *transitions* such that $P \cap T = \emptyset$;
- $B : P \times T \to \mathbb{N}$ is a backward incidence function such that $B(p_i, t_j)$ represents arc weight from t_j to p_i;
- $F : P \times T \to \mathbb{N}$ is a forward incidence function such that $F(p_i, t_j)$ represents arc weight from p_i to t_j;
- $\lambda : T \to Act \cup \{\tau\}$ is a labelling function of a $DTPN$. If $\lambda(t) \in Act$ then t is called observable or external;
- $SI : T \to \mathbb{T} \times \mathbb{T} \cup \{\infty\}$ is a function that associates to each transition a static firing interval;
- $\Gamma : Act \to D$ is a function that associates to each action its static duration.

I is the set of all intervals of a DTPN such that $I(t) = [min, max]$ is the interval associated to the transition t. We denote by $\downarrow I(t) = min$ and $\uparrow I(t) = max$, two functions which give respectively the lower and upper bound of an interval.

As commonly in use in the literature, we write $^\circ t$ (resp. t°) to denote the set of places such that $^\circ t = (p \in P / B(p,t) > 0$ (resp. $t^\circ = \{p \in P / F(p,t) > 0\}$), and $^\circ p$ (resp. p°) to represent the set of transitions such that $^\circ p = \{t \in T / F(p,t) > 0\}$

(resp. $p^\circ = \{t \in T/B(p,t) > 0\}$). Noting that marked DTPN is a tuple $\langle PN, M_0 \rangle$, such that $PN = \langle P, T, B, F, \lambda, SI, \Gamma \rangle$ is a DTPN, and M_0 is its initial marking, where $\forall p \in P : M(p) \in \mathbb{N}$.

2.3 Durational Action Timed Automata

Durational action Timed Automata (daTA) is structurally a subclass of timed automata [14, 15, 30]. However, the difference to be underlined is the one concerning the semantics associated with the model. The daTA model [21, 31] is a timed model defined by a timed transition system based on a true-concurrency semantics, expressing parallel behaviors and supporting at the same time timing constraints, explicit actions duration, structural and temporal non-atomicity of actions i.e., actions may be divisible and of non-null duration.

The daTA model supports the notions of urgency and deadlines as timing constraints of the system. An action duration is expressed by a duration condition associated to states of the model. On the other hand, timing constraints, due to restrictions on the enabling domain of an action, are expressed by the *enabling constraint G* (for Guard) and by *urgency constraint D* (for Deadline) at the level of daTA transitions. In addition, a transition represents only the start of an action, end of execution is captured by the corresponding duration expressed at the level of target state. On the target state, a timed expression manifests that the action is potentially in execution.

From operational point of view, with each action a *clock* is associated which is *reset* at the start of the action. This clock will be used in the construction of the timing constraints as guards of the transitions. Figure 2 b) shows the structure of the corresponding daTA to DTPN of Figure 2 a). This daTA is composed of three states and two transitions labelled with two actions a and b of durations 3 and 2 units of time, respectively. From the initial state S_0 of the illustrative daTA, the execution of action a leads to a reset of clock x associated with it. The expression $x \geq 3$ in state S_2 represents a duration condition on action a and means that a is potentially in execution until the clock x reaches the value 3. The action a does not wait for the end of any other action, so the clock designated by x is used in the enabling domain of this action.This enabling domain will be expressed by the guard and the deadline on the clock x as $(1 \leq x \leq 2)$ and $(x \leq 2)$. Below, we define the timed domain modeling clocks of daTA.

Definition 3. Let $\mathcal{H}$ be a set of clocks with non-negative values (within a time domain $\mathcal{H}$, like $\mathbb{Q}^+$ or $\mathbb{R}^+$). The set $\Phi_t(\mathcal{H})$ of temporal constraints γ over $\mathcal{H}$ is $\gamma x \sim t$, where x is a clock in $\mathcal{H}$, $\sim \in \{=, <, >, \leq, \geq\}$ and $t \in \mathbb{T}$. F_x is used to indicate a constraint of the form $x \sim t$. A valuation v for $\mathcal{H}$ is a function which associates to each $x \in \mathcal{H}$ a value in $\mathbb{T}$. The valuation v for $\mathcal{H}$ satisfies a temporal constraint γ over $\mathcal{H}$ iff γ is true by using clock values given by v. For $I \subseteq \mathcal{H}$, $[I \rightarrow 0]$ indicates the valuation for $\mathcal{H}$ which assigns 0 value to each $x \in I$, and agrees with v over the other clocks of $\mathcal{H}$. The set of all valuations for $\mathcal{H}$ is noted $\Xi(\mathcal{H})$. The

satisfaction relation $\models$ for temporal constraints is defined over the set of valuations for $\mathcal{H}$ by $(v \models x \sim t) \Leftrightarrow (v(x) \sim t)$ such that $v \in \Xi(\mathcal{H})$. $2^{\mathbb{T}}_{fn}$ is used to denote the set of finite subsets of a set $\mathbb{T}$.

Definition 4. A daTA is a tuple $\langle S, L_s, s_0, \mathcal{H}, T_D \rangle$ of the support Act, where:

- S is a finite set of states;
- $L_s : S \to 2^{\Phi_{t(\mathcal{H})}}_{fn}$ is a function which assigns to each state s the set F of ending condition (duration conditions) of actions possibly in execution in s;
- $s_0 \in S$ is the initial state, such that $L_s(s_0) = \emptyset$;
- $\mathcal{H}$ is a finite set of clocks;
- $T_D \subseteq S \times 2^{\Phi_{t(\mathcal{H})}}_{fn} \times 2^{\Phi_{t(\mathcal{H})}}_{fn} \times Act \times \mathcal{H} \times S$ is the set of transitions.

A transition (s, G, D, a, x, s') represents a switch from state s to state s' by starting execution of action a and resetting clock x. G is the corresponding guard, which must be satisfied to fire the transition. D is the corresponding deadline which requires, at the moment of its satisfaction, that action a must occur.

(s, G, D, a, x, s') can be written $s \xrightarrow{G,D,a,x} s'$. Figure 2 b) shows the structure of the corresponding daTA to DTPN of Figure 2 a).

Definition 5. The semantics of a daTA $\mathcal{A} = \langle S, L_s, s_0, \mathcal{H}, T_D \rangle$ is defined by associating with it an infinite transition system $S_{\mathcal{A}}$ over $Act \cup \mathbb{T}$. A state of $S_{\mathcal{A}}$, viewed as a configuration, is a pair $\langle s, v \rangle$ such that s is a state of $\mathcal{A}$ and v is a valuation for $\mathcal{H}$. A configuration $\langle s_0, v_0 \rangle$ is initial if s_0 is the initial state of $\mathcal{A}$ and $\forall x \in \mathcal{H}$, $v_0(x) = 0$. Two types of transitions between $S_{\mathcal{A}}$ configurations are possible, and which correspond respectively to time passing (Rules (1) and (2)) and the launching of a transition from $\mathcal{A}$ (Rule (3)):

$$\frac{d \in \mathbb{T} \quad \forall d' \leq d, v + d' \not\models \mathfrak{D}}{\langle s, v \rangle \xrightarrow{d} \langle s, v + d \rangle}, \tag{1}$$

$$\frac{\varepsilon \in \mathbb{T} \quad v + \varepsilon \models \mathfrak{D} \wedge \varepsilon \in \eta}{\langle s, v \rangle \xrightarrow{\varepsilon} \langle s, v + \varepsilon \rangle}, \tag{2}$$

$$\frac{(s, G, D, a, x, s') \in T_D \quad v \models G}{\langle s, v \rangle \xrightarrow{a} \langle s', [\{x\} \mapsto 0]v \rangle}. \tag{3}$$

In Rule (2), η corresponds to the smallest real quantity of time in which no action occurs [31]. In Rule (3), $\mathfrak{D} = \vee_{i \in I} D_i$, where $\{(s, G_i, D_i, a_i, x, s_i)\}_{i \in I}$ is the set of all transitions stemming from state s. Indeed, whenever a D_i holds, time cannot progress regardless of the other D_i.

In order to guarantee that at least a transition could be drawn starting from a state if time cannot progress any more within this state, the formula $D_i \Rightarrow G_i$ must be satisfied.

Remark 1. For urgency domains, we require that deadline can be only of the form $x \leq t$ or $x < t$.

3 TRANSLATION OF MARTE SEQUENCE DIAGRAMS TO DTPN

This paper proposes a translation of a high-level model written in MARTE SD towards a specific formal time model that is DTPN. Figure 4 gives a general overview of the approach. The approach will follow two main steps. In the first step, MARTE sequence diagrams are formalized. The translation method is developed in the second step. It is defined inductively starting from the basic elements.

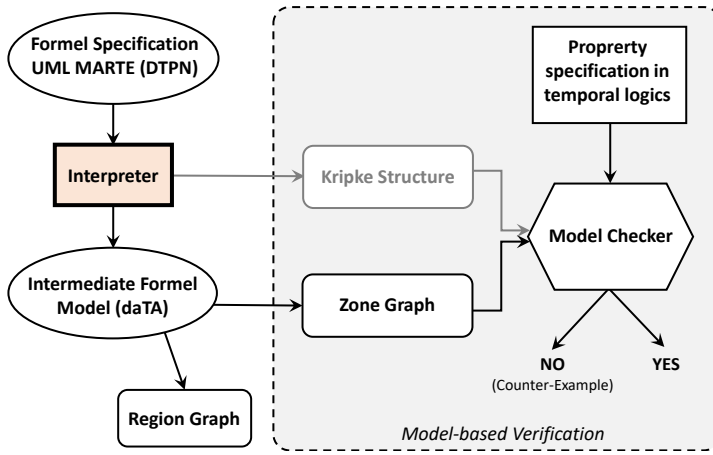


Figure 4. Verification process

3.1 Formalisation

Sequence diagrams are semi-formal notation, in other words the syntax and semantics notations are open to different interpretations. In the literature, there are several research papers that address the formal formalization of sequence diagrams [32, 33, 34, 35, 36, 37, 38, 39, 40, 41]. Among these works, the approach of [41] is an interesting one as a formal translation. In this work, the authors have defined formal rules for the translation of an UML sequence diagram to a corresponding coloured Petri nets. Nevertheless, the transmission processing between two communicating objects in the sequence diagram is specified as a single transition with an abstraction to the different primitives of communication, i.e. sending process, transmitting and receiving process. In addition, this approach models the transmission without further consideration of any details in relation with its execution like *latency* and *time of execution*.

To overcome these limitations and in order to consider timing constraints and duration of execution, we propose a formal translation of MARTE sequence diagrams to Time Petri nets with action duration model. Besides the formal translation and the timing consideration, we investigate the parallel execution specification. For this purpose, the true-concurrency semantics based DTPN model is used to obtain durational action Timed Automata (daTA) (semantics representation) [42, 43]. In daTA, both true concurrency and action duration are considered. This structure allows the verification of properties related to parallel evolution of actions within timing constraints. We note that some properties related to reachability may be checked by means of KRONOS and UPPAAL like tools [43, 44]. However, for properties dealing with true concurrency behaviors, FOCOVE model checker may be used [45].

So, in the proposed approach, we follow the same formalization principle proposed in [41] with some differences. In fact, on one hand the transmission primitives specification between communicating parties is defined and in another hand the formalization is made inductive by associating a DTPN specification to each event and defining the translation of composite interaction in terms of its DTPN corresponding sub-elements. Also, we specify timing constraints, like latency, and execution duration of each action separately in order to model transmission process under timing constraints.

3.2 Formal Definitions

In formal terms, we define the MARTE sequence diagrams as follows:

Definition 6. MARTE $SD = (N, O, E, <_g, M, Act, D, Tc, \lambda, S, T, Pre, Post, Cf)$ where:

- N is a set of diagram names;
- O is a finite set of objects;
- $E = \bigcup_{i \in O} E_i$ is a set of events such that $E_i \bigcap E_j = \emptyset$ for any $i \neq j \in O$;
- $<_g = < \cup \{\bigcup_{i,j \in O, i \neq j} <_{i,j}\}$
  - $<= \bigcup_{i \in O}$ is a set of partial orders on events of E_i such that $\forall i \in O, <_i \subseteq E_i \times E_i$;
  - $<_{i,j}$ defines an order between events e, e' such that $\forall (e, e') \in <_{i,j}$, $e \in O_i$, and $e' \in O_j$ or $e \in O_j$ and $e' \in O_i$ and e precedes e'.
- M is a finite set of message labels;
- *Act* is a set of actions. Each action $a \in Act$ can be launched by event. According to the specification context, we can distinguish different kinds of actions. Examples for actions are *synchronous sending (Sysen) or asynchronous sending a message (Asysen), synchronous receiving (Syrec) or asynchronous receiving*

a message (Asyrec), sending reply (Sreply), Receiving reply (Rreply), transmissions of message, activities or any behavior to be executed in the system.

$$Act = \{Sysen, Asysen, Syrec, Asyrec, Sreply, Rreply, Activity1, \ldots, Activityn, Trans1, \ldots, Transn\};$$

- $Dc : Act \to \mathbb{T}$ is a function that associates to each action a duration which may be null;
- $Tc : E \to \mathbb{T} \times \mathbb{T}$ is a function that associates a time interval to each event such that $\forall e \in E_i, Tc(e) = [d, d + t]$ which means that event e should be executed in the time interval $[d, d + t]$;
- $\lambda : E \to Act$ is a labelling function which associates an action name to each event. For each event $e \in E_i$, e may be defined by 3-uple $\langle e, D(\lambda(e)), Tc(e)\rangle$.
- S is the set of all the possible states with $S = \bigcup_{O_i \in O} S_i$ where S_i is the set of states of an object O_i. $S_i = \bigcup_{e \in E_i} S_e$, such that:
  - $\forall e \in E$, $S_e = \{s_e^0, s_e^1\}$;
  - If $e \in E$ is associated to the sending action then $S_e = \{s_e^0, s_e^1, s_e^{out}\}$;
  - If $e \in E$ is associated to the receiving action then $S_e = \{s_e^0, s_e^1, s_e^{in}\}$.

  Similarly to events, different objects cannot share the states, $S_i \cap S_j = \emptyset$ for $O_i \neq O_j \in O$;
- T is a set of transitions, such that for each transition $t \in T$ is associated to an event $e \in E$ and is of the form $\langle e, D(\lambda(e)), Tc(e)\rangle$. The arcs linking states to transitions are labeled by the Pre function, whereas the arcs linking transitions to states are labeled by the $Post$ function;
- $Pre : N \to 2^{S \times T}$ is an input function associating for each diagram sd, a set of arcs, where $(s, t) \in Pre(sd)$ defines the arc from s to t;
- $Post : N \to 2^{S \times T}$ is an output function associating for each diagram sd, a set of arcs, where $(s, t) \in Post(sd)$ defines the arc from t to s.

  For a given diagram sd:
  - The set of input arcs of transition $t \in T$ is denoted ${}^\circ t = \{(s, t) \in Pre(sd) \mid s \in S\}$;
  - The set of output arcs of transition $t \in T$ is denoted $t^\circ = \{(s, t) \in Post(sd) \mid s \in S\}$;
  - The set of input arcs of state $s \in S$ is denoted ${}^\circ s = \{(s, t) \in Pre(sd) \mid t \in T\}$;
  - The set of output arcs of state $s \in S$ is denoted $s^\circ = \{(s, t) \in Post(sd) \mid t \in T\}$;

- CF is a combined fragment defined by a type of operator and one or more operands.

- *Type* $= \{seq, alt, par, opt, break, loop, ref, strict, critical, neg, assert, ignore, consider\}$;
- *Op* is a finite set of operands;
- *Guard* is a boolean or temporal expression which associates a guard to an operator or to a combined fragment;
- *Frag* is a set of nested combined fragments inside the n^{th} operands of the m^{th} interaction fragments.

3.3 MARTE SD to DTPN Transformation

In the next lines, we present the translation rules and we explain through examples the translation between the input elements of the MARTE sequence diagram to the output elements of DTPN. Considering the different types of transmission between communicating objects, duration of the desired activities to be executed and the timing constraints imposed on the sending and receiving transmitted messages. By using DTPN, the action durations are fixed once and for all at the enabling moment of the system. The case where actions have durations chosen from an interval $[m, M]$ is not considered. Also, we note that the TimeConstraint stereotype presented in Section 2 is expressed by time interval $[min, max]$.

3.3.1 Basic Elements

Definition 7. For a given diagram sd, let $\langle e_1, d_1, tc_1\rangle$ and $\langle e_2, d_2, tc_2\rangle$ be two different transitions corresponding to a given transmission, such that:

- If $O_i, O_j \in O$ and $i \neq j$ such that $e_1 \in E_i$ and $e_2 \in E_j$, then the transitions represent an external evolution between both objects O_i and O_j. It is said an inter-objects transmission;
- If $\exists O_i \in O$ such that $e_1, e_2 \in E_i$ and $e_1 < e_2$, and $\nexists e_3 \in E_i$ such that $e_1 < e_3 < e_2$, then the transitions represent a local evolution in the object O_i. It is said an intra-objects transmission. In such case, $d_1 = d_2 = 0$ and $tc_1 = tc_2 = [0, 0]$.

Inter-objects transmission.

- **Synchronous transmission:** Synchronous transmissions are used when the sender waits for a response from the receiver to continue its operations. A synchronous transmission blocks the progression of the operations of its sender until the receiver gets its message (weak synchronization) or until the sender receives the response (strict synchronization).

  Let us consider the example of Figure 5 which shows a synchronous transmission and its reply. In the following paragraph, we detail how various elements of this sequence diagram (source model) are translated to the elements of DTPN (destination model).

In Figure 5, the Synchronous transmission represents an interaction between two objects O_1 and O_2 associated to both events e_1 and e_2 necessarily different. The translation of this transmission to an equivalent DTPN is based on the interpretation of each event and its associated action in the diagram with taking into account the two states (*before* and *after*) of this event. To construct the DTPN, we split the synchronous transmission in two principal steps:

1. sending the synchronous transmission and
2. receiving the synchronous transmission reply.

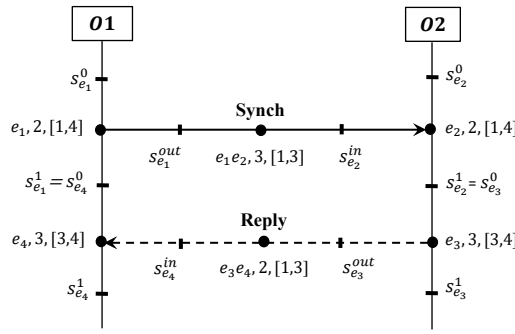


Figure 5. Synchronous transmission

Step 1:

1. Object O_1 launches a transmission sending operation at event e_1 to object O_2, which is constrained by the time interval $[1, 4]$ and duration equal to two units of time. The event e_1 is translated to a Petri net transition t_{e_1} constrained by the time interval $[1, 4]$ and an action of duration 2 units of time. The two states (before and after) of event e_1 are translated to the two places $s^0_{e_1}$, $s^1_{e_1}$. Also, we add the arcs that relate place to transition or transition to place. This translation is illustrated by Figure 6 a).
2. Object O_2 receives the transmission action at event e_2, with duration equal to 2 which may be delayed by 4 units of time. Since, the constraint interval $[1, 4]$ is considered. The corresponding DTPN is represented by Figure 6 b).
3. For representing the intermediate transmission action between object O_1 and object O_2, we create an event named e_1e_2 and two states s^{out}_{e1}, $s^{in}_{e_2}$. The event e_1e_2 models the action, which takes in our case a duration between 4 and 6 units of time. For considering this characteristic, we consider the constraint interval $[1, 3]$ and a duration of the transmission action equal to 3 units of time. Figure 6 c) shows the corresponding DTPN.

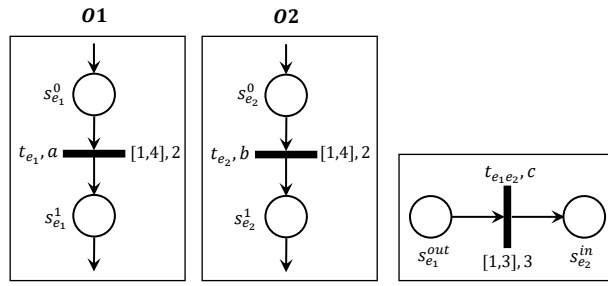


Figure 6. Elementary synchronous transmission corresponding DTPNs

4. After adding the arcs relating the three subnets DTPN models two subnets are related to two subnets, the complete resulting construction of this step is given by Figure 7.

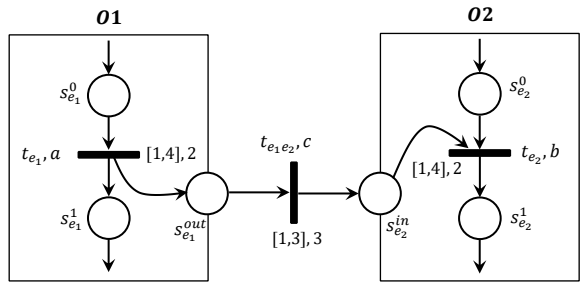


Figure 7. Sending transmission corresponding DTPN

Formally, S and T are the least sets verifying the following construction conditions:

- Let $\{t_{e_1}, t_{e_1e_2}, t_{e_2}\} = \{\langle e_1, 2, [1,4]\rangle, \langle e_1e_2, 3, [1,3]\rangle, \langle e_2, 2, [1,4]\rangle\}$ be a set of transitions modeling a synchronous transmission between objects O_1 and O_2.
- Let $e_1 \in E_1$ and $e_2 \in E_2$ be two events, such that $\lambda(e_1) =$ Sysen and $\lambda(e_2) =$ Syrec, with $(e_1, e_2) \in <_{i,j}$, $\{s^0_{e_1}, s^1_{e_1}, s^{out}_{e_1}, s^0_{e_2}, s^1_{e_2}, s^{in}_{e_2}\} \subseteq S$ and $\{t_{e_1}, t_{e_2}, t_{e_1e_2}\} \subseteq T$, then:

$$
\begin{aligned}
{}^\circ t_{e_1} &= \{s^0_{e_1}\};\\
t^\circ_{e_1} &= \{s^1_{e_1}, s^{out}_{e_1}\};\\
{}^\circ t_{e_2} &= \{s^{in}_{e_2}, s^0_{e_2}\};\\
t^\circ_{e_2} &= \{s^1_{e_2}\};
\end{aligned}
$$

$$^{\circ}t_{e_1e_2} = \{s^{out}_{e_1}\};$$
$$t^{\circ}_{e_1e_2} = \{s^{in}_{e_2}\}.$$

Step 2: Due to its similarity with the sending transmission in step1, it is possible to apply the same translation scheme in this step. As a consequence of the transmission reception by object O_2, this later responses by executing the behavior that is matched to that transmission action.

1. After the transmission processing, object O_2 sends the transmission reply at event e_3 back to object O_1. The event e_3 is characterized by a duration equal to 3 units of time and time interval $[3, 4]$. This is represented by the following DTPN (Figure 8 a)).

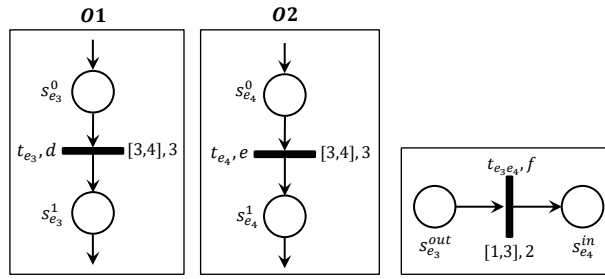


Figure 8. Elementary transmission reply corresponding DTPNs

2. Objects respond to messages that are generated by objects executing communication actions. The object O_1 receives transmission reply at event e_4, which has 3 units of time as duration and $[3, 4]$ as timing constraint. The corresponding DTPN is illustrated by Figure 8 b).
3. For modeling the reply transmission action between sending action at event e_3 and receiving action at event e_4, we create an event named e_3e_4 and two states $s^{out}_{e_3}$, $s^{in}_{e_4}$. This event takes in this case a duration between 3 and 5 units of time. Since then we consider the constraint interval $[1, 3]$ and a duration of the action equal to 2 units of time. The corresponding DTPN model is shown in Figure 8 c).
4. At this stage, the arcs to compose the three subnet DTPNs models are added. The complete resulting model is shown by Figure 9.

At last, the result of translation of the example in Figure 5 is depicted by Figure 10.

Formally, S and T are the least sets verifying the following construction conditions:

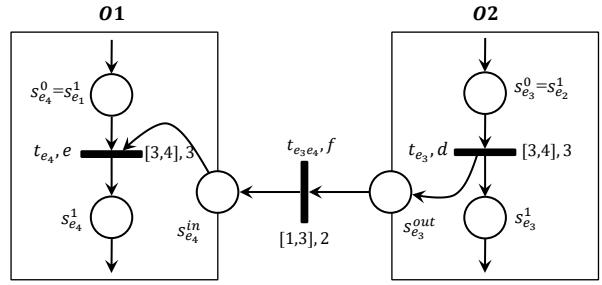


Figure 9. Transmission reply corresponding DTPN

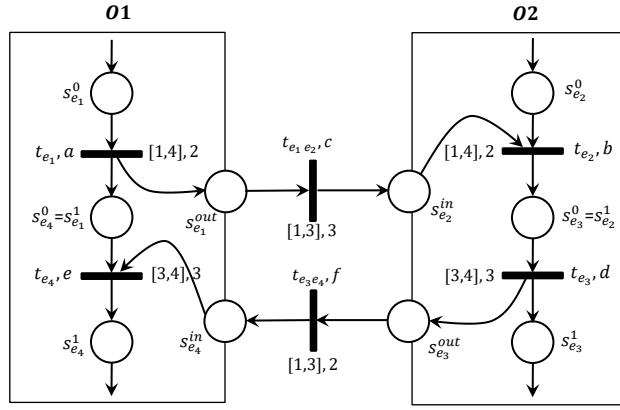


Figure 10. Synchronous transmission corresponding DTPN

– Let $\{t_{e_3}, t_{e_3e_4}, t_{e_4}\} = \{\langle e_3, 3, [3,4]\rangle, \langle e_3e_4, 2, [1,3]\rangle, \langle e_4, 3, [3,4]\rangle\}$ be a set of transitions which models a transmission reply between the two objects O_1 and O_2.
– Let $e_3 \in E_2$ and $e_4 \in E_1$ be two events, such that $\lambda(e_3) =$ Sreply and $\lambda(e_4) =$ Rreply with $(e_3, e_4) \in <_{i,j}, \{s^0_{e_3}, s^1_{e_3}, s^{out}_{e_3}, s^0_{e_4}, s^1_{e_4}, s^{in}_{e_4}\} \subseteq S$ and $\{t_{e_3}, t_{e_4}, t_{e_3e_4}\} \subseteq T$, then:

$$
\begin{aligned}
{}^\circ t_{e_3} &= \{s^1_{e_2}\};\\
t^\circ_{e_3} &= \{s^1_{e_3}, s^{out}_{e_3}\};\\
{}^\circ t_{e_4} &= \{s^{in}_{e_4}, s^1_{e_1}\};\\
t^\circ_{e_4} &= \{s^1_{e_4}\};\\
{}^\circ t_{e_3e_4} &= \{s^{out}_{e_3}\};\\
t^\circ_{e_3e_4} &= \{s^{in}_{e_4}\}.
\end{aligned}
$$

- **Asynchronous transmission:** Asynchronous transmission is used when the sender does not need to wait for response from receiver, it continues its execution after sending the message. The basic functionality is the same as the synchronous transmission. We consider the sequence diagram in Figure 11, which shows the transmission of an asynchronous message between objects O_1 and O_2 at events e and e'.

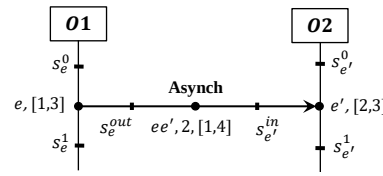


Figure 11. Asynchronous transmission

1. The event e in object O_1, represents the sending action, which is instantaneous and with timing constraint $[1, 3]$. The event e is interpreted as a Petri net transition t_e constrained by the time interval $[1, 3]$, its two states are translated to places s_e^0 and s_e^1. Moreover, we add the arcs linking places to transitions. This translation is represented by Figure 12 a).

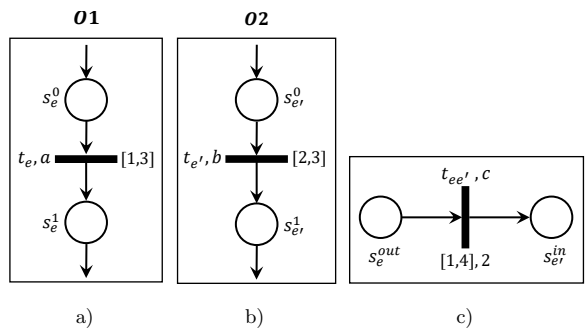


Figure 12. Elementary asynchronous transmission corresponding DTPNs

2. The receiving action in object O_2 is represented by the event e' and the reception message should be in time interval between 2 and 3 units of time. The interpretation of this part can be represented by the DTPN given in Figure 12 b).
3. For representing the asynchronous transmission operation between sending action and receiving action, we add an event named ee' between both events e and e' that takes in our case a duration between 3 and 6 units of time. Hence, we consider the constraint interval $[1, 4]$ and a duration of the action equal to 2 units of time. Figure 12 c) illustrates the corresponding resulting DTPN.

4. To ensure asynchronous transmission between objects O_1 and O_2 we add two arcs, the first arc from transition t_e to input state of transition $t_{ee'}$ and the second arc connects the output state of this last transition to transition $t_{e'}$.

As shown in Figure 13, we obtain the complete DTPN associated to the sequence diagram of Figure 11.

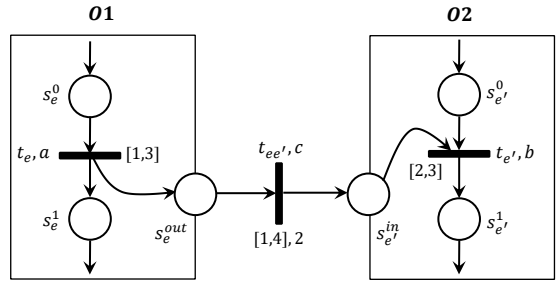


Figure 13. Asynchronous transmission corresponding DTPN

Formally, S and T are the least sets verifying the following construction conditions:

- Let $\{t_e, t_{ee'}, t_{e'}\} = \{\langle e, 0, [1,3]\rangle, \langle ee', 2, [1,4]\rangle, \langle e', 0, [2,3]\rangle\}$ be a set of transitions which models an asynchronous transmission between two objects O_1 and O_2.
- Let $e \in E_1$ and $e' \in E_2$ be two events, such that $\lambda(e) =$ Asysen, $\lambda(e') =$ Asyrec with $(e, e') \in <_{i,j}, \{s_e^0, s_e^1, s_e^{out}, s_{e'}^0, s_{e'}^1, s_{e'}^{in}\} \subseteq S$ and $(t_e, t_{e'}, t_{ee'}) \subseteq T$, then:

$$
\begin{aligned}
{}^\circ t_e &= \{s_e^0\};\\
t_e^\circ &= \{s_e^1, s_e^{out}\};\\
{}^\circ t_{e'} &= \{s_{e'}^{in}, s_{e'}^0\};\\
t_{e'}^\circ &= \{s_{e'}^1\};\\
{}^\circ t_{ee'} &= \{s_e^{out}\};\\
t_{ee'}^\circ &= \{s_{e'}^{in}\}.
\end{aligned}
$$

Intra-objects transmission.
An intra-objects transmission is a self-transmission in sequence diagram where the sender and receiver objects of a message are the same. In this transmission, if two events e and e' belong to the same object O_i, then the event e' immediately follows the event e with no other event in between, Figure 14 a) shows such an example.

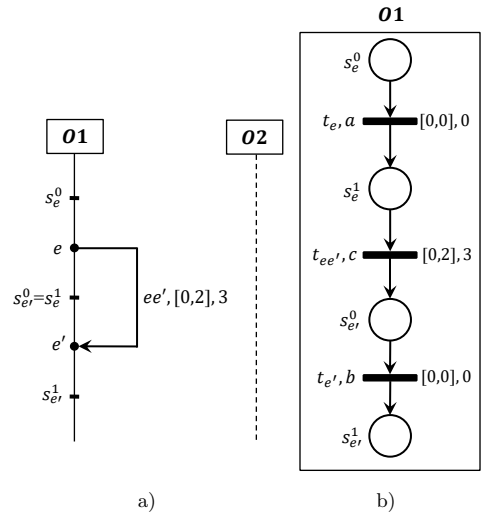


Figure 14. Intra-objects transmission and its corresponding DTPN

In the sequence diagram, sending and receiving actions at events e and e' are instantaneous and without timing constraints, $d1 = d2 = 0$, $ct1 = ct2 = [0, 0]$. The transmission action at event ee' has 3 units of time during the interval $[0, 2]$. In the following, we show how this is mapped to an equivalent DTPN.

1. In object O_1, the sending action at event e is translated to Petri net transition t_e without duration and timing constraint. Both states of this event e are translated to two Petri net places s_e^0 and s_e^1 with two arcs, each one relates a place to the transition t_e. Figure 15 a) shows this construction.

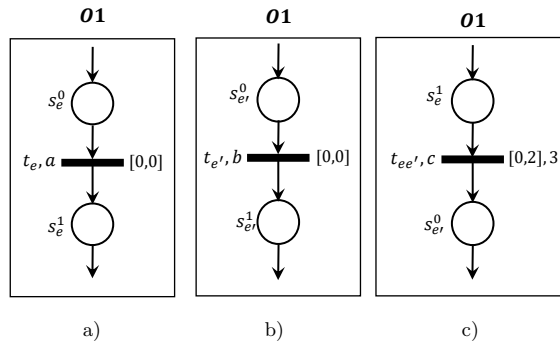


Figure 15. Elementary intra-objects transmission corresponding DTPNs

2. The receiving action at event e' in object O_1 has the same characteristics of the sending action at event e. A similar construction can be made by using two places $s^0_{e'}$ and $s^1_{e'}$, and a transition $t_{e'}$ with two arcs, each one relates a place to the transition $t_{e'}$ (see Figure 15 b)).
3. For modeling the transmission action operation between sending action and receiving action in a same object O_1, we add an event named ee' relating both events e and e' that takes in our case a duration between 3 and 5 units of time. For taking into account this characteristic we consider the constraint interval $[0, 2]$ and a duration of the action associated to the transition $t_{ee'}$ equal to 3. The corresponding DTPN in Figure 15 c). The transmission action at event ee' is translated a Petri net transition $t_{ee'}$ with a duration and a timing constraint. The two places of this transition are the output place s^1_e of transition t_e and the input place $s^0_{e'}$ of transition $t_{e'}$. In addition, two arcs are created, each one relates a place to the transition $t_{ee'}$.
4. For connecting transmission action operation with the two previous parts of intra-objects transmission, we will add two arcs, the first arc from the output place s^1_e of transition t_e to the transition $t_{ee'}$ and the second arc from the transition $t_{ee'}$ to the input place $s^0_{e'}$ of transition $t_{e'}$. The final corresponding DTPN is shown in Figure 14 b).

Formally, S and T are the least sets verifying the following construction conditions:

- Let $\{t_e, t_{ee'}, t_{e'}\} = \{\langle e, 0, [0, 0]\rangle, \langle ee', 3, [0, 2]\rangle, \langle e', 0, [0, 0]\rangle\}$ be a set of transitions modeling intra-objects transmission in O_1.
- Let $e, e' \in E_1$, be two events, such that $\lambda(e) =$ Activity and $\lambda(e') =$ Activity with $\forall(e, e') \in <$, $\{s^0_e, s^1_e, s^0_{e'}, s^1_{e'}\} \subseteq S$ and $\{t_e, t_{e'}, t_{ee'}\} \subseteq T$, then:

$$
\begin{aligned}
{}^\circ t_e &= \{s^0_e\};\\
t_e^\circ &= \{s^1_e\};\\
{}^\circ t_{e'} &= \{s^1_{e'}\};\\
t^\circ_{ee'} &= \{s^1_{e'}\};\\
{}^\circ t_{ee'} &= \{s^1_e\};\\
t^\circ_{ee'} &= \{s^0_{e'}\}.
\end{aligned}
$$

In the different cases of transmission, inter-objects and intra-objects each object in the sequence diagram has a single initial state and a single final state as defined as follows:

- A state s is said an initial iff ${}^\circ s = \emptyset$.
- A state s is said a final iff $s^\circ = \emptyset$.

3.3.2 Combined Fragments

Combined fragments are one important kind of Interactions, which are used to create interactions that are more complex. A combined fragment is defined by an interaction operator with its operands. The operands of a combined fragment can have guards on them as in alternative behavior (alt) and iterative behavior (loop). The guards of the operands are given by boolean expressions (boolean condition or temporal condition).

In UML MARTE profile, we can specify the time constraints on a combined fragment or on an operand using the tag *"execTime"* in the stereotype <<ResourceUsage>>. According to the MARTE SD specification context, the time constraint interval $[a, b]$ associated to this stereotype can take the following values: $\forall a, b \in R^*$

- if the execution of an operand or a combined fragment is urgent then the time interval $[a, b] = [0, 0]$ or $[a, b] = [t, t']$ with $t = t'$;
- if the execution of an operand or a combined fragment is not urgent then the time interval $[a, b] = [0, \infty[$;
- if execution of an operand or a combined fragment is specified with latency then the time interval $[a, b] = [v, v + t]$ with $a \geq 0$ and $a \leq b$.

In this subsection, we give the translation rules of the most used combined fragments such as weak sequencing behavior (seq), strict sequencing behavior (strict), parallel behavior (par), alternative behavior (alt), optional behavior (opt) and iterative behavior (loop). The combined fragment operands are transformed to DTPN subnets using previous translation rules and then integrated to the resulting DTPN.

Weak sequencing combined fragments.
Weak sequencing combined fragments defined by "seq" operator, contain two or more operands. It represent a weak sequencing between the behaviors of its operands. If no other operator is present on a diagram, then weak sequencing should be applied to the Interaction fragments. Figure 16 shows such an example.

1. The initial place $s^0_{f_i}$ with the initial marking is created $M(s^0_{f_i}) = 1$;
2. To synchronize all the operands that will be get involved in the execution of the weak interaction, the transition $t^0_{f_i}$ is created. An arc connecting the initial place $s^0_{f_i}$ to the transition $t^0_{f_i}$ is added;
3. Using the previous translation rules, the two DTPN subnets corresponding to the two operands in weak combined fragments are generated;
4. Since, the purpose of this operator is to allow the execution of only one operand, in other words, the operands are executed in mutual exclusion. In Petri net terms, the transitions related to the operands should be in conflict. So, the place named LP is created and connected to the two initial transitions of the two DTPN subnets;

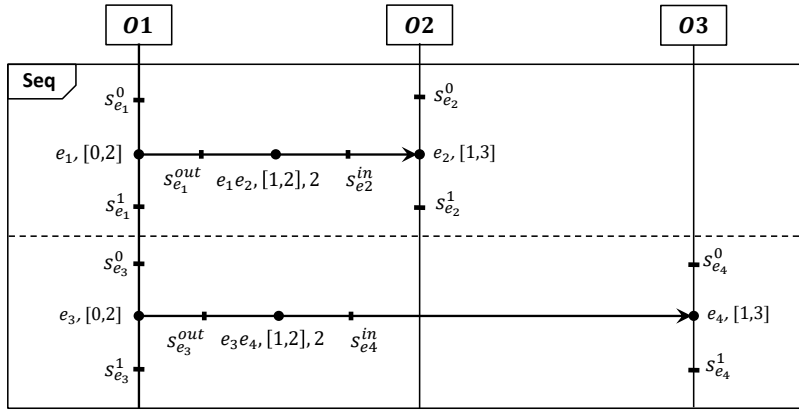


Figure 16. Weak sequencing combined fragments

5. The final transition $t^1_{f_i}$ is created and connected to the last place of every DTPN subnet;

6. The place $s^1_{f_i}$ corresponding to the final place of the DTPN is created and connected, as an output place, to the final transition $t^1_{f_i}$. The equivalent DTPN of weak sequencing combined fragments is given by Figure 17.

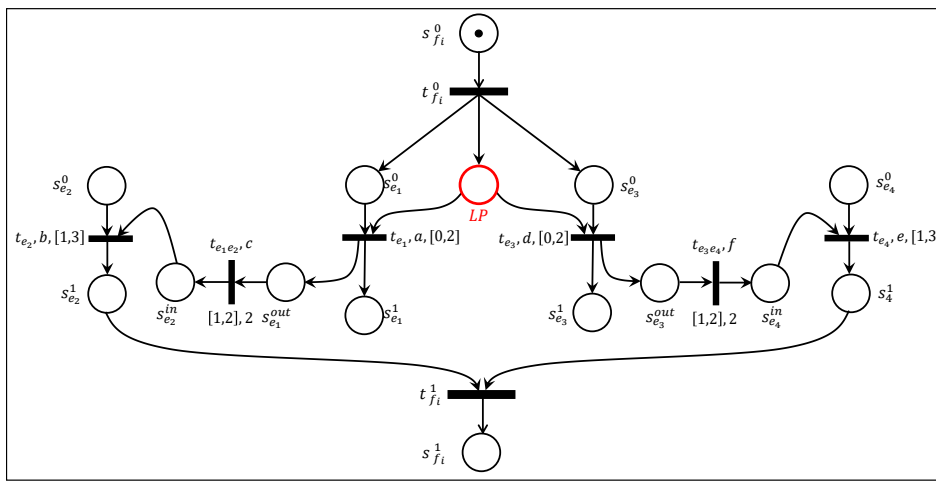


Figure 17. Weak sequencing combined fragments corresponding DTPN

Strict sequencing combined fragments.
Strict sequencing combined fragments are defined by "strict" operator, encloses two or more operands. The operands behaviors must occur in a given order. The order within each operand is preserved. This combined fragments is illustrated by Figure 18.

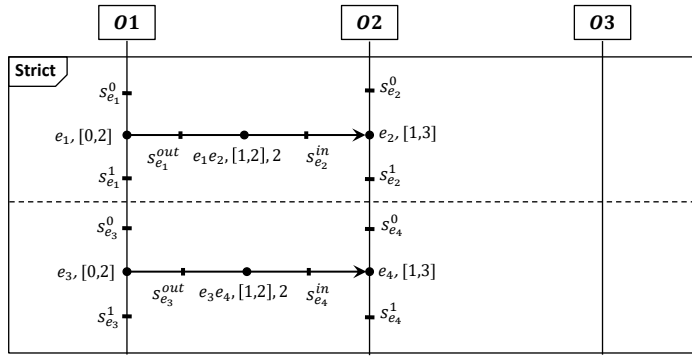


Figure 18. Strict sequencing combined fragments

1. The initial place $s^0_{f_i}$ with the initial marking is created $M(s^0_{f_i}) = 1$;
2. The initial transition $t^0_{f_i}$ is created and connected, as output transition to the initial place $s^0_{f_i}$;
3. Using the translation rules explained in the previous section, the two operands in strict combined fragments are translate to the two DTPN subnets. An arc connecting the last transition of the first DTPN subnet to the initial place of the second DTPN subnet is created;
4. The final transition $t^1_{f_i}$ of the strict combined fragments is generated and an arc connecting the final place of second subnet to the final transition $t^1_{f_i}$ of the strict fragment is added;
5. The final place $s^1_{f_i}$ of strict fragment is created and an arc connecting the transition $t^1_{f_i}$ to the place $s^1_{f_i}$ is added. Figure 19 illustrates the complete construction of the DTPN corresponding to the strict combined fragments.

Parallel combined fragments.
A combined fragment of "par" type is used to specify the concurrent behavior of real-time systems. It describes a parallel execution of the behaviors related to different operands. The behaviors of these operands can be interleaved in any way. However, each operand behavior preserves its predefined order. Figure 20 shows an example of parallel execution of elements *Operand-1* and *Operand-2* with a global time constraint associated to the execution parallel combined fragments. Object O_1

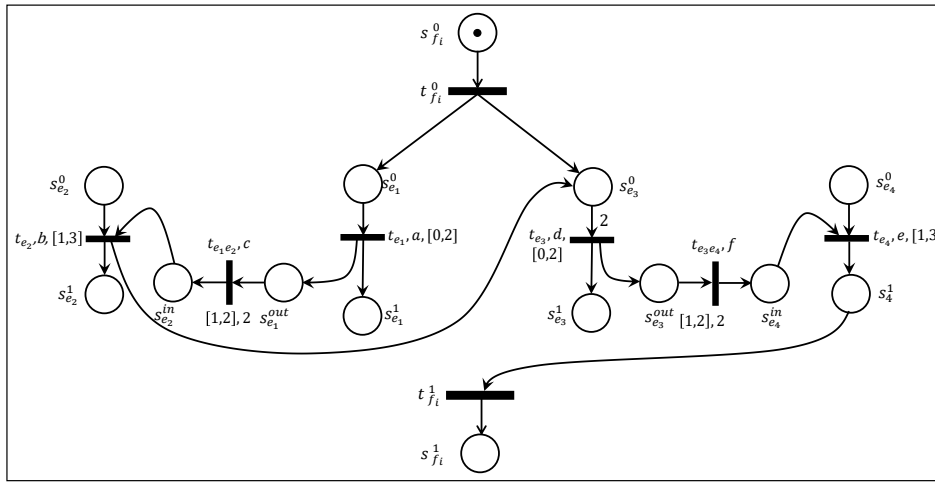


Figure 19. Strict sequencing combined fragments corresponding DTPN

launches concurrently two asynchronous transmission operations. Consequently, the corresponding receiving operations in object O_2 may happen concurrently too. The objects O_1 and O_2 can continue their execution according to the timing constraint of the parallel combined fragment.

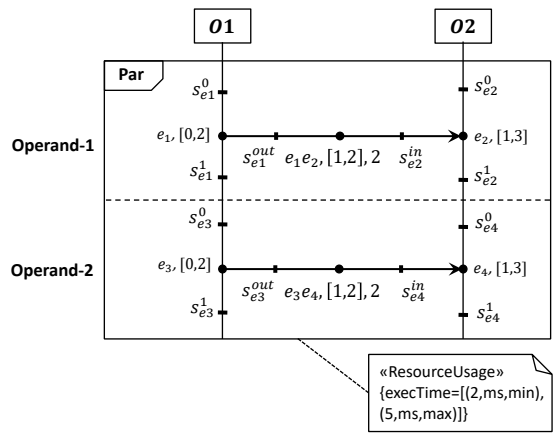


Figure 20. Parallel combined fragments

To detail the translation rules of this fragment type, let us consider the sequence diagram of Figure 20.

1. The initial place $s^0_{f_i}$ with the initial marking is created $M(s^0_{f_i}) = 1$. This marking place is used in the evaluation of the firing condition of the connected transitions;

2. For synchronizing all the operands involved in the parallel combined fragments execution, the transition $t^0_{f_i}$ is created. It is instantaneous and with zeroed time constraint ($d = 0, tc = 0$). An arc connecting the initial place $s^0_{f_i}$ to the transition $t^0_{f_i}$ is then added;
3. Using the translation rules explained in previous section two DTPN subnets $DTPN - 1$ and $DTPN - 2$ are generated, which correspond respectively to *Operand-1* and *Operand-2*. Note that places $s^0_{e_1}$ and $s^0_{e_3}$ have no marking since they become a non initial places ($M(s^0_{e_1}) = 0$, $M(s^0_{e_3}) = 0$). The starting transition $t^0_{f_i}$ is then connected to these places;
4. For synchronizing all operands involved in output of the parallel combined fragments, the final transition $t^1_{f_i}$ is created. It has the time interval $[2, 5]$ as a timing constraint. This transition is connected then, as output transition to each last place of every DTNP subnet ($s^1_{e_2}$, $s^1_{e_4}$). So, the final transition $t^1_{f_i}$ can only be fired when each operand in the combined fragments reaches its final state (one token in its final place);
5. The place $s^1_{f_i}$ corresponding to the final state of the parallel combined fragments is created. The final transition $t^1_{f_i}$ is then connected to this place. The resulting DTPN of this parallel combined fragments is given by Figure 21.

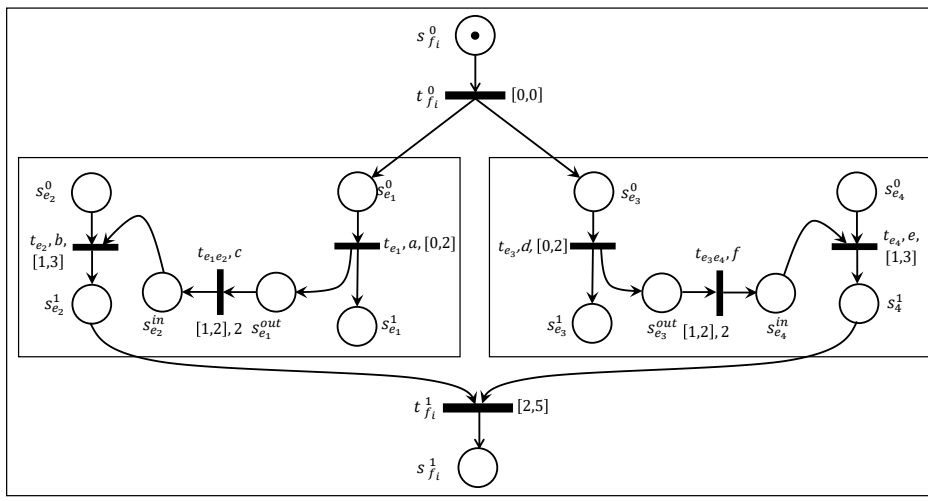


Figure 21. Parallel combined fragments corresponding DTPN

Alternative combined fragments.
Alternative combined fragments defined by "alt" operator, represents a choice of behavior in a fragment. This operator implements a non deterministic choice between

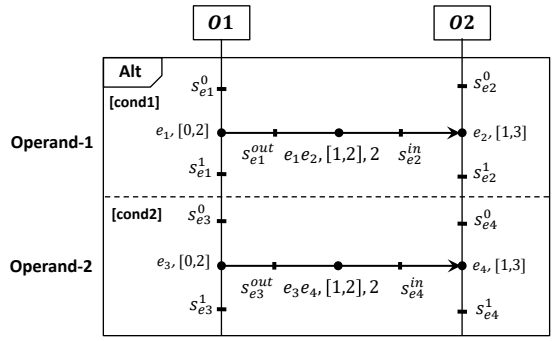


Figure 22. Alternative combined fragments

operands that have their constraints evaluated to true. Figure 22 shows an example of an alternative fragment with two operands.

1. The initial place $s^0_{f_i}$ with the initial marking is created $M(s^0_{f_i}) = 1$;
2. Two conflicting transitions $t^1_{f_i}$ and $t^2_{f_i}$ are generated. In this manner, that only one of the operands is selected, even the transitions guards are both evaluated as true. Two arcs connecting both transitions $t^1_{f_i}$ and $t^2_{f_i}$ to the initial place $s^0_{f_i}$ are added;
3. Using the translation rules explained in the previous section, two DTPN subnets $DTPN-1$ and $DTPN-2$ are generated, which correspond respectively to Operand-1 and Operand-2. The previous transitions $t^1_{f_i}$ and $t^2_{f_i}$ are connected by two arcs to the initials places $s^0_{e_1}$ and $s^0_{e_3}$, respectively, which launch the executions of internal operands (*Operand-1* and *Operand-2*);
4. For representing the output for each DTPN subnet, two final transitions $t^3_{f_i}$ and $t^4_{f_i}$ are generated. These transitions are then related, as an output transitions to the final places $s^1_{e_2}$ and $s^1_{e_4}$, which are associated to sub DTPN-1 and sub DTPN-2, respectively;
5. The final place $s^1_{f_i}$ of DTPN is created and connected to it as the output place to the two final transitions $t^3_{f_i}$ and $t^4_{f_i}$. Figure 23 shows the complete structure of the equivalent DTPN to the alternative combined fragments.

Optional combined fragments.
Optional combined fragments, defined by the operator "opt", contain only one operand which is executed according to a guard condition or a state value. Figure 24 a) gives an example illustrating this operator.

Optional combined fragments is semantically equivalent to the alternative combined fragments. So, it is translated as a simplification of alternative combined fragments with only one operand, considered where the condition evaluation is true. Figure 24 b) shows the corresponding DTPN specification.

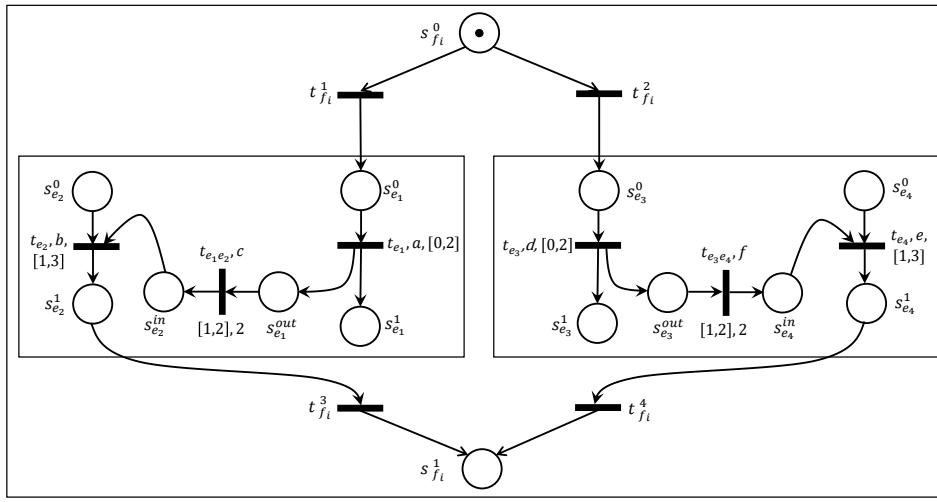


Figure 23. Alternative combined fragments corresponding DTPN

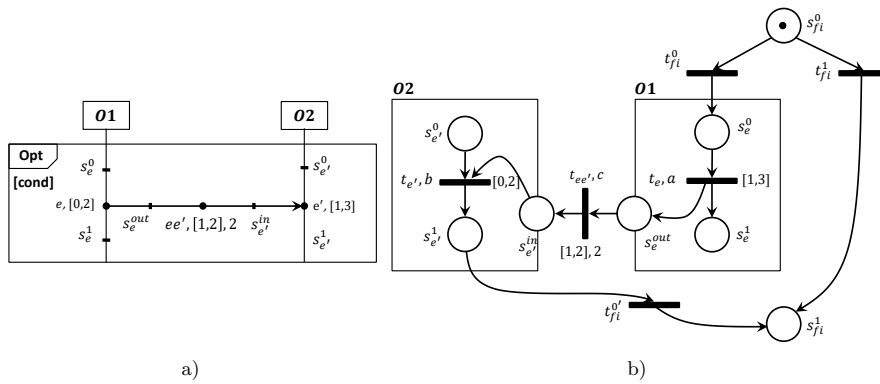


Figure 24. Optional combined fragments and its corresponding DTPN

We can see that in the resulting DTPN, the transition $t^0_{f_i}$ is possible only if a guard condition is true. Otherwise, the transition $t^1_{f_i}$ ends the behavior.

Loop combined fragments.
Iterative combined fragments, denoted by "loop" operator, have only one operand. It defines a recursive behaviour with a guard condition. The guard may either indicate a number of repetitions ($[min, max]$) that should be executed or a boolean expression. Figure 25 a) shows an example representing a loop combined fragments.

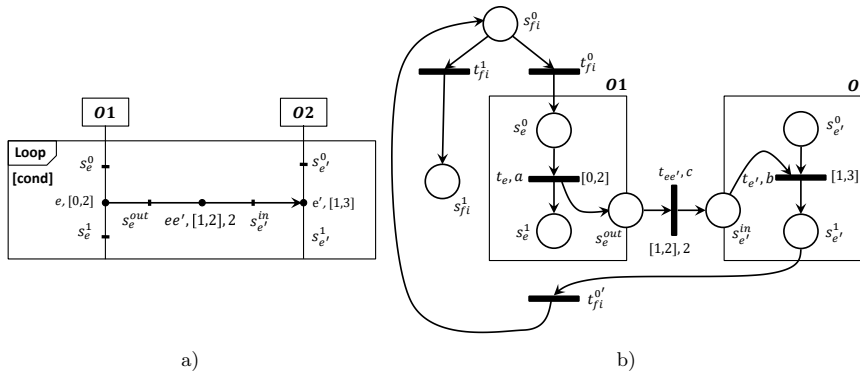


Figure 25. Loop combined fragments and its corresponding DTPN

The translation process follows the following steps:

1. The initial place $s^0_{f_i}$ with the initial marking is created $M(s^0_{f_i}) = 1$;
2. Two initial transitions $t^0_{f_i}$ and $t^1_{f_i}$ are generated. These transitions allow the guard evaluation that serves as a constraint for the combined fragments;
3. Two arcs connecting the initial place $s^0_{f_i}$ to both transitions $t^0_{f_i}$ and $t^1_{f_i}$ are added;
4. Using the translation rules detailed in the previous section, the DTPN subnet corresponding to the loop operand is created;
5. Since, the purpose of this operator is to repeat the execution of the loop operand until the loop guard is evaluated to false, the transition $t^{0'}_{f_i}$ is created and connected to the initial place $s^0_{f_i}$ of the loop fragment. The final place $s^1_{e'}$ of DTPN subnet is then connected to the transition $t^{0'}_{f_i}$;
6. For considering the case when the boolean expression associated to the transition $t^1_{f_i}$ is evaluated to false, the final place $s^1_{f_i}$ is created and connected to the transition $t^1_{f_i}$ as an output place. Figure 25 represents the resulting DTPN structure of the loop combined fragments in Figure 25 b).

4 CASE STUDY

In order to illustrate our approach, for modeling and analysis of real-time embedded systems, we propose to use an interaction fragment of a real case study which is an elevator controller system. We consider two elevators used in a building composed of many floors. The interest of this application is to consider the case of two parallel activities with a non-null duration. These activities correspond to the elevators moves between two different floors at the same time. The elevators are controlled by only one elevator controller system.

Specification.
Assume that two users are at two different floors. At a given instant, each one requests an elevator. For instance, *User-1* is at the first floor while *User-2* is at the seventh floor. It is also assumed that *User-1* wants to go to the seventh floor, and *User-2* wants to go to the first floor.

Since our study focused on parallel activities modeling, we assume that the elevators are parked at floor one and seven, respectively, where their doors are open and the users are inside them. Both users pushed simultaneously the floor button to be visited in the button panel. The elevator controller must move the two elevators in parallel to the requested floors.

To simplify the study, we abstract from elevator system components participating in the interaction which does not affect the study purpose, like *cabin*, *floor sensor* and *door*.

Once the elevator controller has received two requests at the same time, it creates and activates two processes in parallel *Elevator1* and *Elevator2* (two instances). *Elevator1* and *Elevator2* are responsible for the move of the first elevator and the second elevator, respectively.

Two time aspects are taken into consideration, the first one concerns the execution duration of actions and the second one concerns the timing constraints. In the example, the elevator movement action takes 3 units of time ($d = 3\,000$ ms) and the action execution may be delayed and executed in the time interval $[0, 10]$.

Figure 26 shows the sequence diagram representing the time aspects and the interactions between the principal objects involved in the elevator request functionality, with a global time constraint ($tc = [10, 30]$) associated.

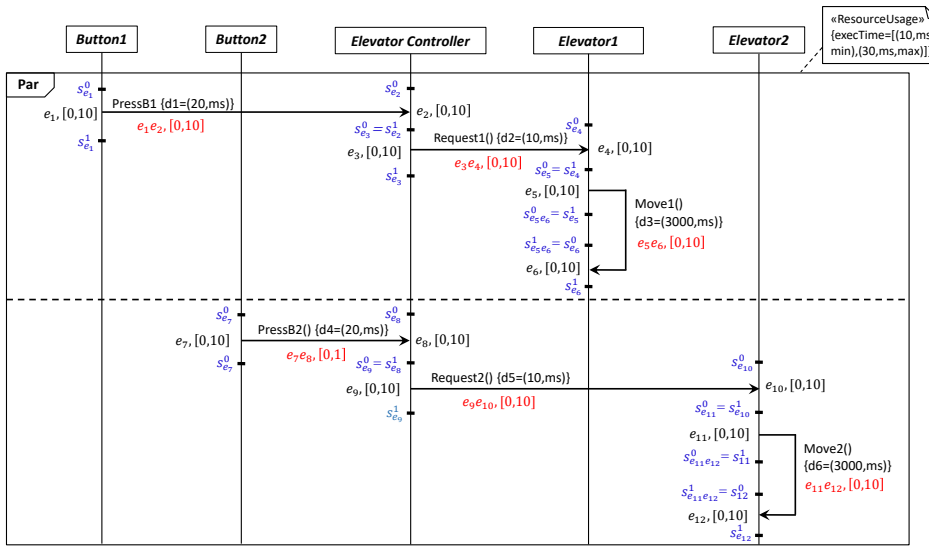


Figure 26. Interactions between objects for the request an elevator

To apply the translation method over the previous sequence diagram, first, we annotate the sending and receiving time events of the transmitted messages between objects. The set of Time events is $E^t = \{e_1, e_2, \ldots, e_{10}, e_{12}\}$. Events representing the intermediate transmissions (colored by red) with their associated duration and timing constraints are also added. Hence, $E^t = \{e_1, e_2, \ldots, e_{11}, e_{12}\} \cup \{e_1e_2, e_3e_4, e_5e_6, \ldots, e_{11}e_{12}\}$. For each object involved in the interaction, two state (colored by blue) to each event on its lifeline is associated, then $S = \{s^0_{e_1}, s^1_{e_1}, s^0_{e_2}, \ldots, s^1_{e_{12}}\}$. Similarly, states are associated to timed events. They correspond to intermediate transmissions in the sequence diagram. Hence, $S = \{s^0_{e_1}, s^1_{e_1}, \ldots, s^0_{e_{12}}, s^1_{e_{12}}\} \cup \{s^{out}_{e_1}, s^{in}_{e_2}, \ldots, s^{in}_{e_{12}}\}$.

The translations method detailed in Section 3 can now be applied on the sequence diagram. Figure 27 depicts the complete construction of the equivalent DTPN.

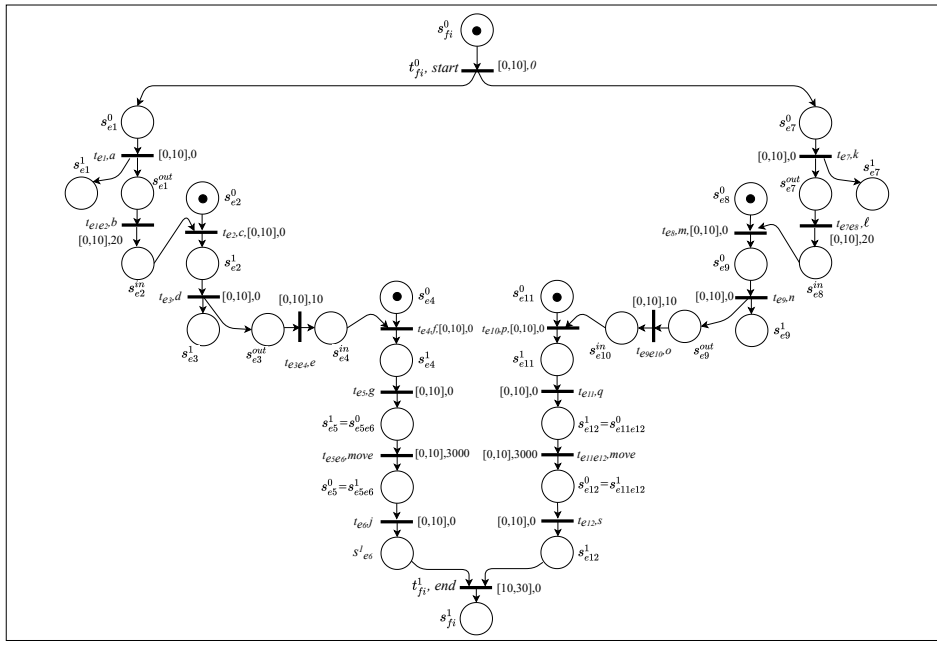


Figure 27. DTPN specification corresponding to sequence diagram for requesting an elevator

According to Figure 27, the initial place of the DTPN is $s^0_{f_i}$ with the initial marking $M(s^0_{f_i}) = 1$. The final place is $s^1_{f_i}$. The initial transition is $t^0_{f_i}$ with time constraint $[0, 0]$. The final transition is $t^1_{f_i}$ with time constraint $[2, 5]$. The places corresponding to the states are $P = \{s^0_{e_1}, s^1_{e_1}, s^0_{e_2}, \ldots, s^1_{e_{12}}, s^{out}_{e_1}, s^{in}_{e_2}, \ldots, s^{in}_{e_{12}}\}$ and the events corresponding to the Petri net transitions are $T = \{t_{e_1}, t_{e_2}, \ldots, t_{e_{11}}, t_{e_{12}}\}$.

In the resulting DTPN, the firing of transitions is bound to a time interval. The transition firing represents action launching which has an explicit duration.

Verification.
To investigate the verification and validation of real time embedded systems, we opt for the operational semantics developed in [20] to generate a semantics model to DTPN specification. The true-concurrency semantics based DTPN model is used to obtain the durational action Timed Automata (daTA) depicted in Figure 28. The resulting daTA has 42 states and 68 transitions; it is generated automatically using a tool integrated into FOCOVE environment [45]. For this reason, we represent only a fragment of the graph. This structure allows the verification of properties related to parallel evolution of actions as shown in the different states of resulting daTA. As an example, state $s36$ is labelled within duration conditions set $\{x \geq 3\,000\,\text{ms}, y \geq 3\,000\,\text{ms}\}$. In this state, actions $Move1$ and $Move2$ can comply in parallel, and each one can finish only if its clock reaches a value equal to its duration.

Note that we have used just two clocks (x and y) to specify all actions of the case study because we have at most two actions in execution at a given time. This is possible due to the dynamic creation of clocks with the reuse of free clocks.

The model checking is mainly based upon the region graph and zone graph algorithms on daTA. The model checking complexity on TA, as daTA, is exponential in the number of clocks. We can observe that using true-concurrency semantics based DTPN, the generated daTA has a lower number of clocks. We can then apply CTL model checking to check some properties.

5 DISCUSSION AND RELATED WORKS

The transformation of MARTE sequence diagrams to formal specifications, for the formal verification, has been investigated in several approaches like [46, 47, 48, 49, 50, 51], but a few approaches as [12, 52, 53] have taken into consideration time specification in the transformation process. Previous works just deal with the flowing of events in sequence diagrams with implicit expression of time and consider only interleaving semantics. On the contrary, we have proposed a translation approach that supports at the same time timing constraints, explicit actions durations, urgency and structural and temporal non-atomicity of actions. Thus, our approach is done to a true concurrency-based real time formal specification models (DTPN and daTA models). The use of daTA's structures as semantics allows firstly, to express concurrent and parallel behaviors in a natural way, i.e. to distinguish between sequential and parallel runs of actions. Therefore, with non-null duration under timing constraints. This is not the case of the interleaving semantics. Secondly, resetting the state clocks which are used to specify time and timing constraints. Therefore, this will lead to reducing the verification time which is often exponential for large size RTES, and to reducing the number of states and transitions in the graph without loss of information, and then escaping from the explosion of the underlying graph.

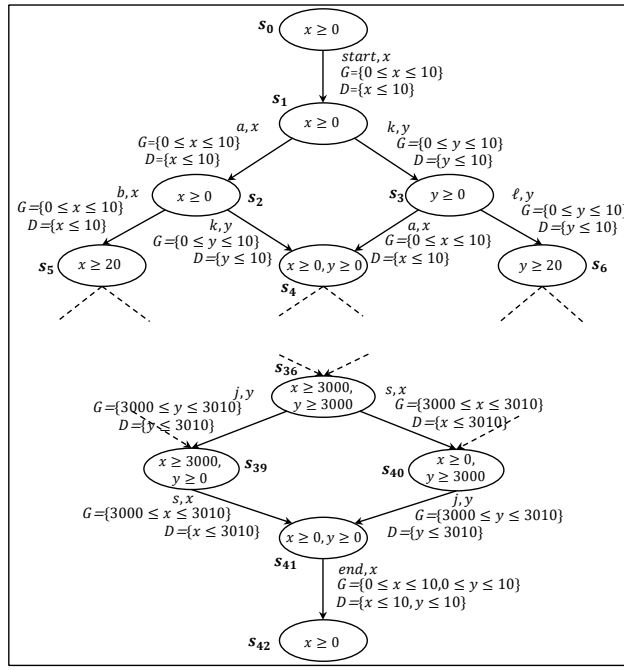


Figure 28. Fragment of daTA corresponding to DTPN

In the proposed method, we borrowed the idea of associating states to events of the sequence diagram as made in [41]. However, the translation method of [41] considers the sequence diagram as a whole entity, in comparison with our translation method, as a set of rules defined to make the translation method inductive. The translation rules start by considering elementary component, which are the events in the sequence diagram. Hence, for each composed component, a translation rule is defined inductively using the translation result of its sub-components. In this manner, a construction of a tool implementing the method may easily be done. As an example, Figure 29 a) shows the asynchronous message events that represent the moments in which the actions send or receive. In terms of the Petri net, each event (e, e', e'') is translated to a transition, an input place and an output place as shown in Figure 29 b).

In [12], authors interpreted parallel activities, modeled in MARTE sequence diagram, by parallel transitions in TCPNIA specification under an interleaving semantics. In this approach, only the execution occurrence duration is modeled. It is specified by a time interval associated to a transition of TCPNIA. In our proposed method, start occurrence, finish occurrence, message occurrence (complete UML name, message occurrence specification), which represent sending and receiving event, and invoking or receiving of operation calls, are considered.

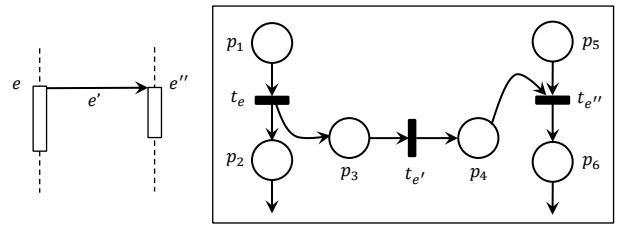


Figure 29. Asynchronous message translating

On the other hand, the proposed works assumed action atomicity hypothesis imposed by the interleaving semantics which handle parallel behaviors as their combined sequential evolution. For more clarification, let us consider the example of Figure 30 b). Using the method proposed in [12], these parallel activities are translated to the TCPNIA specification of Figure 30 c).

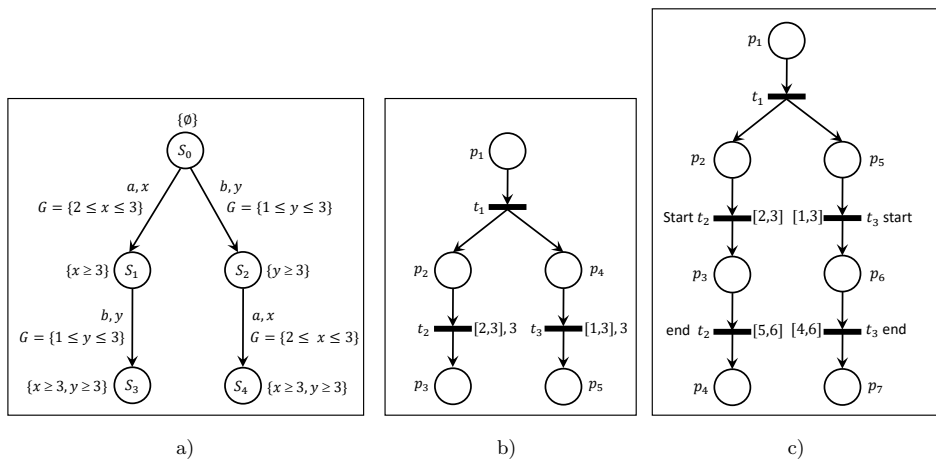


Figure 30. Generating models

For the verification of the required properties, the specification of Figure 30 c) is translated to a Timed Automata. Since the transition execution is instantaneous, there is no way to observe the parallel execution of the two activities. As a solution for such situation, it is possible to interpret each activity having a non-null duration by two sequential transitions modeling the start and the end of the activity. As shown in Figure 30 c), the activity duration is captured in the intermediate state conditioning the execution of the ending transition by the elapsed time. We notice that the composed system may be in the state where the two start transitions are executed before executing the end transitions. Such state captures the parallelism

between the two activities however, such methods augment the number of states, transitions and clocks, which contributes to the state space explosion problem of zone graph corresponding to the Timed Automata specification. As an alternative, the use of DTPN and daTA is an interesting solution (Figure 30 a)).

As a consequence, we remark that the number of transitions of each structure is comparable respectively of those of Figure 30 a). Another advantage concerns the construction of the set of clocks. In our context, a clock is created dynamically during the generation of the semantics models with the reuse of free clocks. On the contrary, other models like Timed Automata, Petri Nets with Deadlines and Time Petri Nets manage [15, 54, 55, 56], at the beginning of modeling, a finite and constant number of clocks recording to the number of actions to execute.

6 CONCLUSION

In this paper, we proposed an operational method for translating MARTE SD specifications to DTPN specifications. As it has been mentioned previously, MARTE SD allows the specification of several kind of behaviors like concurrency, time constraints and action duration. Since, DTPN formal specification model is a true concurrency based semantics, it allows the consideration of the last three behavior characteristics in both syntactic and semantics levels. This latter arguments is the sole motivation of our work. The use of daTA structures as semantics allows properties formal verification, particularly those related to parallel evolution of actions that have non-null duration and under timing constraints. Properties related to reachability may be checked by means of KRONOS and UPPAAL tools, and properties dealing with true concurrency behaviours may be checked using FOCOVE model checker.

In this paper, the translation method has been explained by considering several examples. As for the perspectives of this work, we suggest that it is applied on realistic real-time embedded systems. It could be either integrated to an existing environment system and/or a computer-aided software engineering model checker. It could also be part of a separate formal specification and verification tool. Alternatively, it would be useful to develop a full TCTL model-checker for DTPNs related to MARTE SD specification without passing by Timed Automaton like structures.

REFERENCES

[1] Lee, I.—Leung, J. T.—Son, S.: Handbook of Real-Time and Embedded Systems. New York, Chapman and Hall/CRC, 2008, doi: 10.1201/9781420011746.

[2] Gomes, L.—Fernandes, J. M.: Behavioral Modeling for Embedded Systems and Technologies: Applications for Design and Implementation. Information Science Reference, 2010.

[3] Rayner, M.—Hockey, B. A.—Chatzichrisafis, N.—Farrell, K.: OMG Unified Modeling Language Specification. Version 1.3, © 1999 Object Management Group, Inc., 2005.

[4] Gérard, S.: MARTE: A New Standard for Modeling and Analysis of Real-Time and Embedded Systems. Proceedings of Euromicro Conference on Real-Time Systems (ECRTS '07), Pisa, Italy, 2007.

[5] OMG: UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems, OMG Document Number: Ptc. 2008.

[6] Ge, N.—Pantel, M.—Crégut, X.: Time Properties Dedicated Transformation from UML-MARTE Activity to Time Transition System. ACM SIGSOFT Software Engineering Notes, Vol. 37, 2012, No. 4, pp. 1–8, doi: 10.1145/2237796.2237807.

[7] Ge, M. N.—Cregut, X.: A Framework Dedicated to Time Properties Verification for UML-MARTE Specifications. 2012.

[8] Berthomieu, B.—Vernadat, F.: Time Petri Nets Analysis with TINA. Third International Conference on the Quantitative Evaluation of Systems (QEST '06), 2006, pp. 123–124, doi: 10.1109/QEST.2006.56.

[9] Bošnački, D.—Dams, D.: Integrating Real Time into Spin: A Prototype Implementation. In: Budkowski, S., Cavalli, A., Najm, E. (Eds.): Formal Description Techniques and Protocol Specification, Testing and Verification (PSTV 1998, FORTE 1998). Springer, Boston, MA, IFIP – The International Federation for Information Processing, Vol. 6, 1998, pp. 423–438, doi: 10.1007/978-0-387-35394-4_26.

[10] Bošnački, D.—Dams, D.: Discrete-Time Promela and Spin. In: Ravn, A. P., Rischel, H. (Eds.): Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT 1998). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1486, 1998, pp. 307–310, doi: 10.1007/bfb0055359.

[11] Holzmann, G. J.: The Model Checker SPIN. IEEE Transactions on Software Engineering, Vol. 23, 1997, No. 5, pp. 279–295, doi: 10.1109/32.588521.

[12] Yang, N.—Yu, H.—Sun, H.—Qian, Z.: Modeling UML Sequence Diagrams Using Extended Petri Nets. Telecommunication Systems, Vol. 51, 2012, No. 2-3, pp. 147-158, doi: 10.1007/s11235-011-9424-5.

[13] Yang, N.-H.—Yu, H.-Q.: Modeling and Verification of Embedded Systems Using Timed Colored Petri Net with Inhibitor Arcs. Journal of East China University of Science and Technology, Vol. 36, 2010, No. 3, pp. 411–417.

[14] Alur, R.—Dill, D.: Automata for Modeling Real-Time Systems. In: Paterson, M. S. (Ed.): Automata, Languages, and Programming (ICALP 1990). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 443, 1990, pp. 322–335, doi: 10.1007/bfb0032042.

[15] Alur, R.—Dill, D. L.: A Theory of Timed Automata. Theoretical Computer Science, Vol. 126, 1994, No. 2, pp. 183–235, doi: 10.1016/0304-3975(94)90010-8.

[16] McMillan, K. L.: Symbolic Model Checking. Kluwer Academic Publishers, 1993, doi: 10.1007/978-1-4615-3190-6.

[17] Courcoubetis, C.—Yannakakis, M.: Minimum and Maximum Delay Problems in Real-Time Systems. Formal Methods in System Design, Vol. 1, 1992, No. 4, pp. 385–415, doi: 10.1007/bf00709157.

[18] Gardey, G.—Roux, O. H.—Roux, O. F.: State Space Computation and Analysis of Time Petri Nets. Theory and Practice of Logic Programming, Vol. 6, 2006, No. 3, pp. 301–320, doi: 10.1017/s147106840600264x.

[19] Bouyer, P.—Fahrenberg, U.—Larsen, K. G.—Markey, N.—Ouaknine, J.—Worrell, J.: Model Checking Real-Time Systems. In: Clarke, E., Henzinger, T., Veith, H., Bloem, R. (Eds.): Handbook of Model Checking. Springer, Cham, 2018, pp. 1001–1046, doi: 10.1007/978-3-319-10575-8_29.

[20] Belala, N.—Saidouni, D. E.—Boukharrou, R.—Chaouche, A. C.—Seraoui, A.—Chachoua, A.: Time Petri Nets with Action Duration: A True Concurrency Real-Time Model. International Journal of Embedded and Real-Time Communication Systems (IJERTCS), Vol. 4, 2013, No. 2, pp. 62–83, doi: 10.4018/jertcs.2013040104.

[21] Saidouni, D. E.—Belala, N.: Actions Duration in Timed Models. International Arab Conference on Information Technology (ACIT), 2006.

[22] Micskei, Z.—Waeselynck, H.: UML 2.0 Sequence Diagrams' Semantics. LAAS Technical Report No. 08389, 37, 2008.

[23] OMG: UML Profile for Schedulability. Performance, and Time Specification, Vol. 1, 2005.

[24] Siegel, J. M.: Model Driven Architecture (MDA), MDA Guide Rev. 2.0. Object Management Group, Tech. Rep. ORMSC/14-06-0, 2014.

[25] OMG Unified Modeling Language™ (OMG UML). 2013.

[26] André, C.: MARTE Time and Time Constraints Models and Their Applications. 2011.

[27] Merlin, P.: A Study of the Recoverability of Computer Systems. Ph.D. Thesis, Computer Science Department, University of California, 1974.

[28] Ramchandani, C.: Analysis of Asynchronous Concurrent Systems by Petri Nets (No. MAC-TR-120). Massachusetts Institute of Technology, Cambridge Project Mac, 1974.

[29] Sifakis, J.: Use of Petri Nets for Performance Evaluation in Measuring Modelling and Evaluating Computer Systems. 1977.

[30] Alur, R.: Timed Automata. In: Halbwachs, N., Peled, D. (Eds.): Computer Aided Verification (CAV 1999). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1633, 1999, pp. 8–22, doi: 10.1007/3-540-48683-6_3.

[31] Belala, N.: Modèles de Temps et Leur Intérêt à la Vérification Formelle des Systèmes Temps-Réel. Doctoral Dissertation. 2010, doi: 10.13140/RG.2.2.25932.21129.

[32] Störrle, H.: Trace Semantics of Interactions in UML 2.0. Journal of Visual Languages and Computing, 2004.

[33] Cavarra, A.—Küster-Filipe, J.: Formalizing Liveness-Enriched Sequence Diagrams Using ASMs. In: Zimmermann, W., Thalheim, B. (Eds.): Abstract State Machines 2004. Advances in Theory and Practice (ASM 2004). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3052, 2004, pp. 62–77, doi: 10.1007/978-3-540-24773-9_6.

[34] Cengarle, M. V.—Knapp, A.: UML 2.0 Interactions: Semantics and Refinement. Proceedings of the 3rd International Workshop Critical Systems Development with UML (CSDUML'04), 2004, pp. 85–99.

[35] Eichner, C.—Fleischhack, H.—Meyer, R.—Schrimpf, U.—Stehno, C.: Compositional Semantics for UML 2.0 Sequence Diagrams Using Petri Nets. In: Prinz, A., Reed, R., Reed, J. (Eds.): SDL 2005: Model Driven. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3530, 2005, pp. 133–148, doi: 10.1007/11506843_9.

[36] Haugen, Ø.—Husa, K. E.—Runde, R. K.—Stølen, K.: STAIRS Towards Formal Design with Sequence Diagrams. Software and Systems Modeling, Vol. 4, 2005, No. 4, Art. No. 355, doi: 10.1007/s10270-005-0087-0.

[37] Küster-Filipe, J.: Modelling Concurrent Interactions. Theoretical Computer Science, Vol. 351, 2006, No. 2, pp. 203–220, doi: 10.1016/j.tcs.2005.09.068.

[38] Hammal, Y.: Branching Time Semantics for UML 2.0 Sequence Diagrams. In: Najm, E., Pradat-Peyre, J. F., Donzeau-Gouge, V. V. (Eds.): Formal Techniques for Networked and Distributed Systems (FORTE 2006). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4229, 2006, pp. 259–274, doi: 10.1007/11888116_20.

[39] Fernandes, J. M.—Tjell, S.—Jorgensen, J. B.—Ribeiro, O.: Designing Tool Support for Translating Use Cases and UML 2.0 Sequence Diagrams into a Coloured Petri Net. Sixth International Workshop on Scenarios and State Machines (SCESM'07: ICSE Workshops 2007), IEEE, 2007, doi: 10.1109/scesm.2007.1.

[40] Harel, D.—Maoz, S.: Assert and Negate Revisited: Modal Semantics for UML Sequence Diagrams. Software and Systems Modeling, Vol. 7, 2008, No. 2, pp. 237–252, doi: 10.1007/s10270-007-0054-z.

[41] Bowles, J.—Meedeniya, D.: Formal Transformation from Sequence Diagrams to Coloured Petri Nets. 2010 Asia Pacific Software Engineering Conference, IEEE, 2010, pp. 216–225, doi: 10.1109/apsec.2010.33.

[42] Bouneb, M.—Saidouni, D. E.—Ilie, J. M.: Hierarchical System Design Using Refinable Recursive Petri Net. Computing and Informatics, Vol. 37, 2018, No. 3, pp. 635–655, doi: 10.4149/cai_2018_3_635.

[43] Daws, C.—Olivero, A.—Tripakis, S.—Yovine, S.: The Tool KRONOS. In: Alur, R., Henzinger, T. A., Sontag, E. D. (Eds.): Hybrid Systems III (HS 1995). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1066, 1995, pp. 208–219, doi: 10.1007/bfb0020947.

[44] Larsen, K. G.—Pettersson, P.—Yi, W.: UPPAAL in a Nutshell. International Journal on Software Tools for Technology Transfer, Vol. 1, 1997, No. 1-2, pp. 134–152, doi: 10.1007/s100090050010.

[45] Saïdouni, D. E.—Benamira, A.—Belala, N.—Arfi, F.: FOCOVE: Formal Concurrency Verification Environment for Complex Systems. AIP Conference Proceedings, Vol. 1019, 2008, pp. 375–380, doi: 10.1063/1.2953008.

[46] Ejnioui, A.—Otero, C. E.—Qureshi, A. A.: Formal Semantics of Interactions in Sequence Diagrams for Embedded Software. 2013 IEEE Conference on Open Systems (ICOS), Kuching, Malaysia, 2013, pp. 106–111, doi: 10.1109/icos.2013.6735057.

[47] Saputra, A. B.—Basuki, T. A.—Tirtawangsa, J.: Transformation of UML 2.0 Sequence Diagram into Coloured Petri Nets. 2014 International Conference of Advanced Informatics: Concept, Theory and Application (ICAICTA), IEEE, 2014, pp. 243–248, doi: 10.1109/icaicta.2014.7005948.

[48] Meedeniya, D.—Bowles, J.—Perera, I.: SD2CPN: A Model Transformation Tool for Software Design Models. 2014 International Computer Science and Engineering Conference (ICSEC), IEEE, 2014, pp. 354–359, doi: 10.1109/icsec.2014.6978222.

[49] Zafar, N. A.: Formal Specification and Verification of Few Combined Fragments of UML Sequence Diagram. Arabian Journal for Science and Engineering, Vol. 41, 2016, No. 8, pp. 2975–2986, doi: 10.1007/s13369-015-1999-9.

[50] Meedeniya, D.—Perera, I.—Bowles, J.: Tool Support for Transforming Unified Modelling Language Sequence Diagram to Coloured Petri Nets. Maejo International Journal of Science and Technology, Vol. 10, 2016, No. 3, pp. 272–283.

[51] Soares, J. A. C.—Lima, B.—Faria, J. P.: Automatic Model Transformation from UML Sequence Diagrams to Coloured Petri Nets. Proceedings of the 6th International Conference on Model-Driven Engineering and Software Development – Volume 1: AMARETTO, 2018, pp. 668–679, doi: 10.5220/0006731806680679.

[52] Menad, N.—Dhaussy, P.—Drey, Z.—Mekki, R.: Towards a Transformation Approach of Timed UML MARTE Specifications for Observer-Based Formal Verification. Computing and Informatics, Vol. 35, 2016, No. 2, pp. 338–368.

[53] Andrade, V. C.—Peres, L. M.—Del Fabro, M. D.: Handling Global and Local Time and Energy Constraints of Sequence Diagrams. 2018 UKSim-AMSS 20th International Conference on Computer Modelling and Simulation (UKSim), IEEE, 2018, pp. 73–78, doi: 10.1109/uksim.2018.00025.

[54] Laroussinie, F.—Markey, N.—Schnoebelen, P.: Model Checking Timed Automata with One or Two Clocks. In: Gardner, P., Yoshida, N. (Eds.): CONCUR 2004 – Concurrency Theory. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3170, 2004, pp. 387–401, doi: 10.1007/978-3-540-28644-8_25.

[55] Cassez, F.—Roux, O. H.: Structural Translation from Time Petri Nets to Timed Automata. Journal of Systems and Software, Vol. 79, 2006, No. 10, pp. 1456–1468, doi: 10.1016/j.jss.2005.12.021.

[56] D'Aprile, D.—Donatelli, S.—Sangnier, A.—Sproston, J.: From Time Petri Nets to Timed Automata: An Untimed Approach. In: Grumberg, O., Huth, M. (Eds.): Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4424, 2007, pp. 216–230, doi: 10.1007/978-3-540-71209-1_18.

Nadia Chabbat received her B.Eng. degree from the University of Badji Mokhtar, Annaba, Algeria in 2005. In July 2014, she received her M.Sc. degree in computer science from the University of Badji Mokhtar, Annaba, Algeria. Her research domain is formal specification and verification of real-time embedded systems.

Djamel Eddine Saidouni received his B.Eng. degree from the University of Mentouri, Constantine, Algeria, in 1990. He received his Ph.D. in theoretical computer science from the University of Paul Sabatier, Toulouse, France in 1996. His domain research is the formal specification and verification of complex distributed and real time systems.

Radja Boukharrou is currently Associate Professor in the Faculty of New Technologies of Information and Communication at University of Constantine 2, Algeria. She holds her Ph.D. in computer science from Constantine 2 University. Her current research interests include formal modeling and verification, security and privacy in IoT systems, blockchain technology.

Salim Ghanemi graduated as Computer Science Engineer in June 1981 from Constantine University, Algeria. In December 1982, he received his Master degree in computer science from Aston University, Birmingham, England. In November 1987, he publicly discussed his Ph.D. research in the parallel and distributed programming at Loughborough University, Loughborough, England. From September 1988 till now, he has assumed several teaching and supervising research positions at many universities: Badji Mokhtar University, Annaba, Algeria, Philadelphia University, Amman, Jordan and King Saud University at Riyadh, Kingdom of Saudi Arabia. He has many scientific publications on several topics. In his main research focus is parallel programming, image processing, real time processing and formal verification. At present, he is the Head of a research team working on parallel processing on SoC, a project group attached to the Embedded Systems Laboratory at Badji Mokhtar Annaba, LASE. He occupied several administrative duties such as the Head of the Computer Science Department and the Vice Dean of the Engineering Science Faculty.

Computing and Informatics, Vol. 39, 2020, 1061–1081, doi: 10.31577/cai_2020_5_1061

MOVING TARGET DETECTION BASED ON AN ADAPTIVE LOW-RANK SPARSE DECOMPOSITION

Jiang Chong

School of Information Science and Engineering
Hunan Women's University
410004 Changsha, China
e-mail: jessiejch@qq.com

Abstract. For the exact detection of moving targets in video processing, an adaptive low-rank sparse decomposition algorithm is proposed in this paper. In the paper's algorithm, the background model and the solved frame vector are first used to construct an augmented matrix, then robust principal component analysis (RPCA) is used to perform a low-rank sparse decomposition on the enhanced augmented matrix. The separated low-rank part and sparse noise correspond to the background and motion foreground of the video frame, respectively, the incremental singular value decomposition method and the current background vector are used to update the background model. The experimental results show that the algorithm can deal with complex scenes such as light changes and background motion better, and the algorithm's delay and memory consumption can be reduced effectively.

Keywords: Detection of moving objects, low-rank, sparse decomposition, adaptive robust, principal component analysis

Mathematics Subject Classification 2010: 68U10

1 INTRODUCTION

The important source of people's knowledge is from image information in the world. In many occasions, the transmitted information in images is richer, truer and more specific than other forms of information. The cooperation between the human eye

and the brain enables people to acquire, process, and understand visual information. Humans use vision to perceive external environmental information efficiently. In fact, according to statistics of some foreign scholars, about 80 % of the external information, which is obtained by humans, comes from images which are taken by the eyes. If computers are to be intelligent, as the main carrier for humans to obtain external information, our vision must be able to process image information. Especially in recent years, image data processing with large capacity (such as graphics, images and video) has been widely used in medical, transportation and industrial automation fields.

Most images in nature are constantly changing simulated images. In daily life, the moving targets in these images are often more concerned about us, such as pedestrians, vehicles, and other moving objects. Moving object detection is a popular direction in computer vision and digital image processing, and it is widely used in robot navigation, intelligent video surveillance, industrial detection, aerospace and many other fields. Therefore, moving object detection has become a research hotspot in theory and application in recent years. Moving object detect is an important branch of image processing and computer vision, and it is also a core part of intelligent monitoring systems. The purpose is how to quickly and accurately detect moving targets in surveillance video, that is, moving targets are extracted from sequence images.

The detection of moving targets is the most basic and very important step in video processing. The accurate detection of moving targets is of great significance for tracking, classifying, understanding and analyzing the behavior of subsequent moving targets. In recent years, many scholars have conducted many studies on the detection of moving targets. However, the detection of moving targets faces many challenges, such as multi-mode background interference, lighting changes, and camera shake. They are still research hotspots and difficulties in the field of computer vision.

The representative algorithms include optical flow method, frame difference method and background modeling method in the research of moving target detection. The optical flow method can detect and track moving targets without prior knowledge of the background, but there are problems such as high complexity of the algorithm and sensitivity to illumination changes. In the frame difference method, the moving target contour is obtained by performing differential operations on two adjacent frames in the video image sequence. The algorithm is simple and easy to implement, but it depends on the selected inter-frame time interval. The idea of the background modeling method is to establish a background model, and the video frame is detected with the background model to determine the motion foreground. Compared with the optical flow method and the frame difference method, it has the advantages of less calculation, faster speed, and higher precision. The core of the background modeling method is how to build a model, how to update the model properly to deal with the changes of the background itself.

At present, although there is a large number of moving object detection algorithms, due to the complexity and variability of the actual environment, these

algorithms are not all very robust. The faced problems and difficulties can be summarized as follows:

1. Model initialization problem: During the background initialization training period, because high-quality background models have not been obtained, it often leads to false detection of moving targets;
2. Camouflage phenomenon: Some moving targets may be extremely similar to the background, it may cause the moving targets to be indistinguishable from the background;
3. Illumination change: It is divided into sudden change and gradation of light. The background model should be able to adapt to the gradual change of light in the outdoor environment during the day; correspondingly, the background model can also be adapted to the indoor environment where the lights are suddenly turned on. In short, the change of light will strongly affect the background model, which will most likely cause false detection;
4. Foreground hole phenomenon: When the moving target has a large number of regions with the same color, the internal changes of these regions may lead to inaccurate detection, some internal regions of the foreground are misjudged as the background;
5. Dynamic background: The most common is the leaf shake, of course, water ripples, small target shake;
6. Suddenly stopped moving targets: After some moving objects enter the scene, they stop in the scene. Obviously, the moving target in this case should be identified as the background;
7. Shadow: The shadow of the moving target and the original shadow of the background area can be detected;
8. Noise interference: Noise interference basically belongs to the low data quality, it is caused by the video image which is transmitted or compressed by the webcam;
9. Camera shake: Under some conditions, wind will cause camera shake;
10. Camera self-adjustment: At present, many cameras have automatic control functions, such as lighting control, white balance, and zoom-in and zoom-out functions.

2 RELATED WORKS

Computer vision applications based on videos often require the detection of moving objects in their first steps. Background subtraction is then applied in order to separate the background and the foreground. Background subtraction is surely among the most investigated field in computer vision providing a big amount of publications. Most of them concern the application of mathematical and machine learning models to be more robust to the challenges met in videos. However, the

ultimate goal is that the background subtraction methods developed in research could be employed in real applications like traffic surveillance. There is often a gap between the current methods used in real applications and the current methods in fundamental research. In addition, the videos evaluated in large-scale datasets are not exhaustive in the way that they only covered a part of the complete spectrum of the challenges met in real applications [1].

Background modeling is a technique for extracting moving objects in video sequences. The different foreground sgmentation methods are categorized based on the way of obtaining the reference frame. The different approaches can be categorized into basic modeling, statistical modeling, clustering algorithms, methods based on fuzzy modeling, neural and neuro-fuzzy methods, etc.

Neural network methods train networks for a specific amount of video frames, allowing it to update its weights in order to model the background. Recently, the most common background subtraction methods are based on deep neural networks. The convolutional Neural Networks (CNN) was introduced to perform the segmentation task [2]. The benefit of this approach is that foreground objects segmentation can be achieved independently to the temporal characteristics. CNN can achieve the segmentation with only spatial characteristics because the incoming examples are independent. The pertinent features are selected using CNNs and transmitted into a classifier to segment moving objects. Finally, a median filter and or a fully connected framework are applied. Lim et al. introduced an approach based on a triplet CNN for a multistage background feature embedding using an encoder-decoder model [3]. A pre-trained convolutional network, VGG-16 Net is adapted under a triplet framework in the encoder part to incorporate an image in multiple scales into the feature space. Only a few training samples are used. The network takes an RGB image in three different scales and generates a foreground segmentation probability mask for the corresponding image. In the period of 2018–2019, numerous deep learning models either based on auto-encoder [4, 5, 6] and CNNs [7, 37, 9, 10, 11] have been proposed. However, all these methods are supervised and have been trained on ground truth video frames of datasets and tested on the same types of videos.

The representative work of the background modeling method is as follows: Gaussian mixed model (GMM) was proposed by Stauffer et al. [12], multiple Gaussian models were used to fit the multi-peak state of pixel brightness changes, and it has better illumination robustness. The non-parametric kernel density estimation (KDE) was proposed by Elgammal et al. [13], it does not need to assume the type of the probability density function in advance, and multiple background samples were used to estimate the probability density of pixels. The algorithm can effectively suppress shadows, but there exists a large cache size, difficulty in selecting core bandwidth, etc. The codebook model were used proposed by Kim et al. [14], CodeBook of a multiple codeword was established for each pixel, multi-modal background can be detected in real time, there is good global illumination robustness. However, the memory consumption of the algorithm is large. The self-organizing background subtraction (SOBS) was proposed by Maddalena et al. [15, 16], it draws

on the characteristics of neural networks, a pixel in the background model is mapped to multiple locations in the model. The relevance of pixel neighbors' domain space is used in the updating method. The VIBE (visual background extractor) modeling method was proposed by Barnich et al. [17], a random sampling strategy was used to initialize the background model, and the background model is updated with a second random sampling method. This airspace information propagation mechanism enables the algorithm to deal with camera shake, the real-time performance of the algorithm is also high. Pixelmann adaptive segmentation (PBAS) was proposed by Hofmann et al. [18], the idea of cybernetics is introduced. The foreground judgment threshold and background model update rate change adaptively, there is high recognition accuracy in the algorithm, but it takes more time to calculate and set multiple thresholds. In short, these algorithms based on statistical models treat each pixel as an independent, unrelated individual, without excavating the inherent nature of high-dimensional data.

In recent years, compressive sensing has received a great deal of attention as a new signal sampling theory [19], and it has been widely used in data compression, pattern recognition, and wireless communication. Compressed sensing is a signal sparse in a certain transform domain. The signal is sampled at a sampling rate much lower than the Nyquist frequency. The observation matrix is used to linearly project the high-dimensional sparse signal to a low-dimensional subspace. By solving the optimization problem, the original signal with high probability is reconstructed from the subspace. A low-rank representation can be seen as a generalization of compressed sensing in two-dimensional data [20]. The rank of the matrix is used as a sparse measure. Compared with the traditional subspace learning model, sparse representation has better robustness to data containing outliers and strong noise. In sparse representation, the robust principal component analysis (RPCA) is also known as low rank and sparse matrix decomposition[21, 22], it mainly considers how to recover low rank data from observations that are subject to large sparse noise pollution observations. Taking into account the low rank of the background and the sparseness of the motion foreground, Candes et al. first applied the low-rank sparse decomposition to background modeling [23]. In this method, the moving target and the background area can be separated effectively from the monitoring video sequence without an independent training stage. The premise of the algorithm is to assume that the background is almost or completely stationary. Subsequent researchers also launched a series of studies based on this method of moving target detection algorithms. DECOLOR algorithm combines RPCA with motion recognition [24] and it combines Markov random field (MRF) a priori. Noise and small background motion can be eliminated effectively in the smoothness of MRF, but the foreground area will be "Smooth" and result in the loss of details of the moving target. Gao et al. proposed the block sparse RPCA [25]. First the low-rank sparse decomposition of the sampled samples was performed to obtain the outlines block, and then the second decomposition was performed to obtain the moving target. Liu et al. introduced the concept of structured sparse in considering the spatial correlation of outliers, and low-rank sparse decomposition is used with different regularization parameters for

each pixel group [26]. Bouwmans et al. summarized and evaluated the technique of decomposition into Low-rank [27].

When RPCA is applied to moving target recognition, the above algorithms are improved from different perspectives. However, they all have the following problems:

1. All vectors of the video are vectorized and loaded into memory once, when the number of video frames is relatively more, it will cause memory overflow;
2. They do not take into account complex changes such as changes in light, background motion in the actual video surveillance environment, and poor robustness.

3 RPCA APPLIED TO BACKGROUND SUBTRACTION A SHORT OVERVIEW

3.1 Matrix Approaches

In many practical applications, the given data matrix D tends to be low-rank or approximately low-rank, the random amplitude is arbitrarily large, but the error of sparse distribution destroys the low rank of the original data. Therefore, matrix D can be decomposed into the sum of two matrices, that is, $D = A + E$, where matrices A and E are unknown, and A is a low-rank matrix. When the noise obeys an independent and identically distributed Gaussian distribution, the classical principal component analysis (PCA) can be used to obtain the optimal matrix A [28], that is, the following optimization problem is solved:

$$\min ||D - A|| \quad \text{s.t.} \quad \text{rank}(A) \leq k \tag{1}$$

wherein, $\text{rank}(\cdot)$ represents the rank of the matrix, and $||\cdot||$ represents the 2 norm. Classical PCA can be effectively solved by singular value decomposition, but it is limited to cases where noise N_0 is small and independent of Gaussian distribution. Candes et al. proposed RPCA to solve the problem of E, it is sparse and large noise [23]. That is, matrix E in the above model is a sparse matrix with arbitrary amplitude. The RPCA model is shown in Equation (2), where $||\cdot||_0$ represents 0 norm.

$$\min(\text{rank}(A, ||E||_0) \quad \text{s.t.} \quad D = A + E. \tag{2}$$

Introducing the regularization parameter λ, this bi-objective optimization problem is transformed into a single-objective optimization problem:

$$\min \text{rank}(A) + \lambda ||E||_0 \quad \text{s.t.} \quad D = A + E. \tag{3}$$

In Equations (2) and (3), the objective functions includes $\text{rank}(A)$ and $||E||_0$, it is non-linear non-convex combined optimization functions, and its solution is an NP-Hard problem. Since the matrix norm $||\cdot||_*$ of the matrix is the envelope of its rank,

the 0 norm and the 1 norm of the matrix can be equivalent under certain conditions, so the convex relaxation of the matrix is the following optimization problem:

$$\min ||A||_* + \lambda ||E||_1 \quad \text{s.t.} \quad D = A + E. \tag{4}$$

3.2 Tensor Approaches

The convex optimization problem shown in Equation (3) is also called principal component pursuit (PCP). Candes et al. prove that as long as the singular vector distribution of the low-rank matrix A is reasonable and the non-zero elements of the sparse matrix are evenly distributed, then the PCP can recover the original low-rank matrix A from any unknown error with a probability close to unity [23].

Robust Principal Component Analysis (RPCA) is a modification of the widely used statistical procedure of principal component analysis (PCA) which works well with respect to grossly corrupted observations. A number of different approaches exist for Robust PCA, including an idealized version of Robust PCA, which aims to recover a low-rank matrix A from highly corrupted measurements $D = A + E$ [23]. RPCA can be applied to video surveillance, face recognition, collaborative filtering and other aspects. In video surveillance, each frame of the video is vectorized and successively arranged into a matrix. The stable background part corresponds to the low-rank part of the matrix, and the motion foreground can be used as a sparse noise part. Therefore, PCP can be used to detect moving targets in a smooth background.

This decomposition in low-rank and sparse matrices can be achieved by techniques such as Principal Component Pursuit method (PCP) [23], Stable PCP [29], Quantized PCP [30], Block based PCP [31], and Local PCP [32]. Then, optimization methods are used such as the Augmented Lagrange Multiplier Method (ALM [33]), Alternating Direction Method (ADM [34]), Fast Alternating Minimization (FAM [35]) or Iteratively Reweighted Least Squares (IRLS [36]), Real-time Robust Principal Components Pursuit [37], Memory-efficient dynamic robust PCA [38], Incremental Principal Component Pursuit [39], Spatio-temporal Sparse Subspace Clustering [40], Double-constrained RPCA [41], ARF and OR-PCA Background Subtraction [42].

In addition to the matrix segmentation technique used in target detection, there are Tensors Decomposition techniques. In mathematics, tensors are geometric objects that describe linear relations between geometric vectors, scalars, and other tensors. Elementary examples of such relations include the dot product, the cross product, and linear maps. Geometric vectors, often used in physics and engineering applications, and scalars themselves are also tensors. A more sophisticated example is the Cauchy stress tensor T, which takes a direction v as input and produces the stress T(v) on the surface normal to this vector for output, thus expressing a relationship between these two vector. With Tensors Decomposition as target detection, people study some effective methods, such as Outlier-Robust Tensor PCA [43], Tensor Based Low-Rank and Saliently Fused-Sparse Decomposition [44], Online Stochas-

tic Tensor Decomposition [45], Stochastic Decomposition into Low Rank and Sparse Tensor [46].

4 ADAPTIVE LOW RANK SPARSE DECOMPOSITION ALGORITHM

Compared with the moving object detection algorithm based on pixel background modeling, the advantage of RPCA is that it can separate the moving foreground area from the background area effectively without a separate training stage. But the premise of the algorithm is to assume that the background is almost or completely static. However, this assumption is difficult to establish in most of the real monitoring scenarios. And Equation (4) is used to accurately separate the foreground and the background, you need to load a larger number of video frames. If the number of frames is too small, slow moving objects are recognized as part of the background. Since the time complexity of the singular value decomposition is $O(m^3)$ $(D \in R^{m\times n})$ in the RPCA solution, the time cost of the algorithm rises in a cubic order with the increase of m, and loading and solving a large number of video frames cause that a low Rank-sparse decomposition has high latency in moving target recognition.

Based on this, an adaptive low-rank sparse decomposition algorithm is proposed in this paper. Through the partial loading and iterative updating of the video frame, the memory occupancy can be reduced while the accuracy of the detection effect can be effectively guaranteed, and the motion target detection delay can be reduced. The algorithm framework is shown in Figure 1.

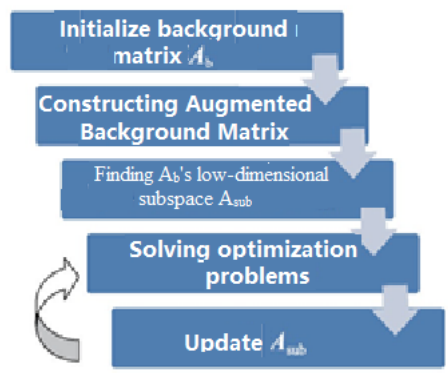


Figure 1. Algorithm framework

Robust PCA considers such a problem: the general data matrix D contains structural information and also contains noise. Then this matrix can be decomposed into two matrices $D = A + E$, A is low rank (due to a certain amount of structural

information inside causing a linear correlation between rows or columns), E is sparse (containing noise, here is the foreground (moving image) , it is sparse). Robust PCA is to decompose a matrix D into a matrix with as low a rank as possible and a matrix as sparse as possible. D consists of original image sequence frames, its each column is an image pixel. A_b is the column subset of A (image background). D consists of original image sequence frames, its each column is an image pixel. A_b is the column subset of A (image background). The implementation of the algorithm in Figure 1 is described below.

4.1 Dynamic Designer Motion Simulation

In order to reduce delay of the algorithm, we consider processing a small number of video frames at a time. However, in order to ensure the accuracy of the motion foreground and background separation, A still needs a large number of columns to guarantee the low rank of $||\cdot||_*$. Therefore, an augmented matrix is composed of the calculated background vector and the background vector to be calculated [47].

If the background vector of the first k frames of video has been obtained, assuming that b_j represents the background vector of the j^{th} frame of video, where $j \in [1, k]$,n is the total number of pixels per frame, the following background matrix is obtained: $A_b = [b_1, b_2, \ldots, b_k] \in R^{n\times k}$.

The augmented background matrix $\widehat{A}$ is composed of A_b and A, where $A \in R^{n\times m}$ is the background matrix of m new frames to be calculated: $\widehat{A} = [A_b, A] \in R^{n\times(k+m)}$.

Therefore, $||A||_*$ in the model is shown in Equation (4), it can be replaced by $||\widehat{A}||_*$.

4.2 Low Dimensional Background Modeling and Model Solving

Since the time complexity of optimizing the augmented low-rank matrix $\widehat{A}$ is $O(k + m)^3$, as the new frame is continuously processed, its time cost increases in a cubic order. Therefore, it is very necessary to perform dimensionality reduction on A_b, that is to project A_b into a low-dimensional subspace. Let this subspace be $A_{sub} \in R^{n\times p}$ in order, a new augmented matrix $[A_{sub}, A] \in R^{n\times(p+m)}(p \ll k)$ is obtained. In other words, the goal is to find an optimal solution A_sub, that makes the nuclear norm of $[A_{sub}, A_{new}]$ closest to $\widehat{A}_{aug}$, the optimization problem of Equation (5) is obtained:

$$A_{sub} = \arg\min |||\widehat{A}||_* - ||[A_{sub}, A_{new}]||_*|. \tag{5}$$

First, Singular Value Decomposition (SVD) is performed on A_b, the main features of the high dimensional matrix is extracted to construct A_{sub}:

$$A_b = UDV^T. \tag{6}$$

In Equation (6), $D \in R^{n\times k}$ is a diagonal matrix which is composed of singular values of A_b; $U \in R^{n\times n}$ is a matrix which is composed of left singular vectors;

$V \in R^{k\times k}$ is a matrix which is composed of right singular vectors. Selecting the first P largest singular values of D, it constitutes a diagonal matrix $D_p \in p \times p$, and selecting the top P column of U, it constitutes a matrix $U_p \in n \times p$, which results in a low-dimensional subspace A_{sub}:

$$A_{sub} = U_p D_p. \tag{7}$$

The variables of the model (4) is replaced to obtain the model (8):

$$\min ||[A_{sub}, A]||_* + \lambda||E||_1 \quad \text{s.t.} \quad D = A + E. \tag{8}$$

Considering the efficiency of the implementation, the inexact Lagrange multiplier method (inexact ALM) is used to solve the above optimization problem [33, 34]. In Equation (8), the augmented matrix of the kernel norm term consists of known and unknown columns. It is necessary to introduce a new variable to replace the augmented matrix, and the variable splitting method is used to separate the objective function into: $Z = [A_{sub}, A]$.

Augmented Lagrangian functions are built for the above optimization problem to be solved:

$$L(Z, E, Y) = ||Z||_* + \lambda||E||_1 + < Y, D - Z - E > +\mu||D - Z - E||_F^2/2$$

where Y is a Lagrangian multiplier; $\mu||D - Z - E||_F^2/2$ is a penalty function term, and μ is a penalty function factor. Repeat the iteration of the following update formula until convergence:

$$Z_{k+1} = \arg\min_Z L(Z, E_{k+1}, Y_k, \mu_k) = D_{1/\mu_k}(D - Z_{k+1} + Y_k/\mu_k),$$

$$E_{k+1} = \arg\min_E L(Z_{k+1}, E, Y_k, \mu_k) = S_{1/\mu_k}(D - E_{k+1} + Y_k/\mu_k).$$

Among them, $S_\tau(x)$ is a soft threshold operator, $S_\tau(x) = sgn(x) \times max(|x| - \tau, 0)$; $D_\tau(x)$ is a singular value threshold operator, $D_\tau(x) = US_\tau(\Sigma)V_*$ ($X = U\Sigma V_*$ indicates singular value decomposition for X). The convergence condition is $||D - Z - E||_F \le \sigma||D||_F$, where $\sigma = 10^{-7}$ [24].

4.3 Background Model Updates

In order to obtain a new background model, it is used to process the next batch of frames, after each calculation, it is necessary to use the currently obtained A to update A_b. This article uses the incremental SVD method to update A_b [48].

$$A_b \approx U_p D_p V_p^T, [U^{new}, D^{new}] = svd([\omega_b A_b, \omega_a A])$$

$$\approx svd[\omega_b U_p D_p V_p^T, \omega_a A] = svd[\omega_b U_p D_p, \omega_a A] = svd[\omega_b A_{sub}, \omega_a A].$$

A new background model A_b^{new} is obtained:

$$A_{sub}^{new} = U_p^{new} D_p^{new} \tag{9}$$

werein, U_p^{new} is composed of the first P columns of U^{new}; D_p^{new} is a diagonal matrix which are composed of P largest singular values. ω_a and ω_b are the weights of the control updating speed (ω_a, $\omega_b \in [0, 1]$). The larger ω_a, the faster the updating of the background model; the larger ω_b, the slower the updating of the background model, so the multimode scenes with complex backgrounds can choose $\omega_a > \omega_b$. At the same time, the matrix V does not need to be calculated in the above formula operation, which effectively reduces the time complexity when updating the model.

5 EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

The experimental simulation hardware environment is: Intel Core i3 dual-core processor, clocked at 3.4 GHz, and main memory is 4 GB. The software environment is: Window10 operating system, Matlab R2018a.

5.1 Experiments on CDnet and Wallflower Datasets

The test data selects the standard video libraries CDnet and Wallflower for motion target detection. CDnet 2014 includes 11 series of basic scenes, dynamic backgrounds, camera shakes, etc. Wallflower includes 7 scenes such as motion background and lighting gradient. At the same time, the data set also gives the groundtruth of each frame segmentation foreground, which facilitates the comparative analysis of the algorithm. Two scenes, Running and Highway, were selected. The following algorithm was compared with the GMM algorithm, the PCP algorithm, and the DECOLOR algorithm. Figure 2 shows the motion foreground recognition effect of each algorithm.

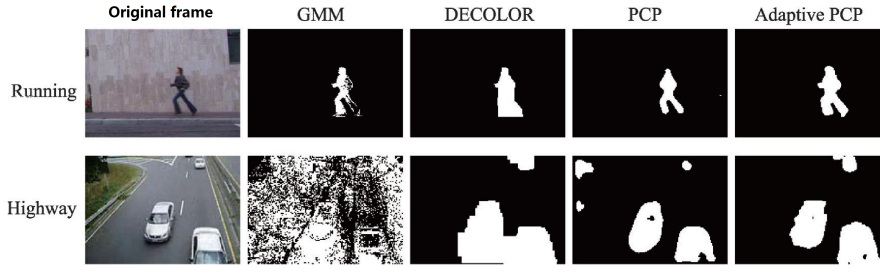


Figure 2. Experiment results of 4 algorithms in different scenes

The running scene of the first row in Figure 2 is a single-moving target. The motion scene in a simple background gives the experimental results of the 17$^{\text{th}}$ frame.

The GMM algorithm uses multiple Gaussian models to fit the changes in pixel values. The identified motion foreground contours are more detailed, but there are holes in the right and left arms of the motion object. The DECOLOR algorithm uses the MRF to remove noise and small background motions, but it makes the motion foreground smooth and loses contour details. Both the PCP and the method of this paper can completely identify the motion prospects. The second line Highway is a complex scene with multiple moving objects and multi-modal backgrounds coexisting with light changes. The branches are shaken in the upper left corner, that are the sports background. In the figure, the video frame 76 is taken in the truncated light. Obviously, the speed of updating the background model of GMM cannot adapt to the sudden change of light, leading to a large area of misjudgment. The DECOLOR algorithm is robust to sudden changes in light, but due to the loss of contour details, it leads to adhesion of the two motion foregrounds. Although the PCP algorithm can accurately identify all the moving targets, it mistakenly identifies the branches that are sloshing in the upper-left corner as motion foreground, because the PCP algorithm assumes that the background is completely stationary. The method in this paper is able to cope with sudden light changes and multi-mode background disturbances, and it has the best recognition effect in this scene. Because the inexact Lagrangian multiplier method is used for optimization, for the running scenes with relatively simple background and foreground, the algorithm can achieve convergence with 33 iterations, and the complex scene with multiple moving targets has a complex background. It takes 71 iterations to achieve convergence. That is, the simpler the scene, the faster the algorithm converges.

In quantitative analysis of the algorithm, the indicators are recall (Re), accuracy (precision, Pr) and comprehensive performance (F-measure, F1). Among them, TP (true positive) indicates true positive, that is, the number of pixels are correctly detected as the front sight; FN (false negative) indicates false negative, that is, the number of pixels are detected as background points by mistake; FP (false positive) indicates false positive, that is, the number of pixels are detected as the previous sighting mistake.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Table 1 gives a quantitative comparison of the performance parameters of each algorithm in the two scenarios. It can be seen that the introduction of an adaptive process in the PCP algorithm significantly improves the ability of the algorithm to handle complex backgrounds.

Algorithm	Scene	Re	Pr	F1
GMM	Running	0.69	0.82	0.87
	Highway	0.62	0.42	0.5
DECOLOR	Running	0.82	0.76	0.79
	Highway	0.83	0.71	0.77
PCP	Running	0.89	**0.88**	**088**
	Highway	0.87	0.79	0.83
Adaptive PCP	Running	**0.9**	0.87	0.88
	Highway	**0.9**	**0.88**	**0.89**

Table 1. Comparison of average performance

Note: GMM is Gaussian mixed model; PCP is principal component pursuit; The entire process of DECOLOR is detecting contiguous outliers in the low-rank representation.

Figure 3 compares the time efficiency of PCP and Adaptive PCP. Experimental results show that by loading part of the video frame and singular value decomposition of the background model, the delay of the algorithm is effectively reduced. At the same time, since the PCP algorithm needs to load all the frames at once. Compared to the Running ($180 \times 144 \times 42$ frames) scene, Highway scenes with a low resolution but a large total number of frames ($160 \times 112 \times 119$ frames) require more average processing time. While Adaptive PCP loads the video frames to be calculated in batches, the average processing time of a single frame is inversely proportional to the resolution of the video.

5.2 Experiments on the VOT2014 Dataset

In order to intuitively compare the experimental results of the algorithm, the two running and highway modes with no lower resolution and fewer frames are selected. If we choose the PETS2006 scene of the dataset ($320 \times 240 \times 1\,200$ frames), the memory overflow will occur in the traditional PCP algorithm, and the algorithm can still obtain the detection result. Figure 4 shows the foreground detection result of the 740^{th} frame of the scenario.

The algorithm has been implemented. We have tested the algorithm with both simulated and actual sequences of images of vehicles in different landscapes. The system can detect and track targets in real-time. We achieved the frame rate of 4 frames/second for detection and 15 frames/second for tracking of 100×90 pixels target in 352×288 pixels video frames. We tested the proposed algorithm with wide variety of image sequence. Figure 5 shows some results for detection of various objects in arbitrary background. As it is shown the algorithm has successfully detected the targets. In Figure 6 the tracking results for a vehicle are shown. In this example, the tracked vehicle turns over the road and its shape and size change. As it shown in the pictures both the size and the shape of the vehicle varies but our tracking algorithm can successfully track it. Results showed the accuracy of

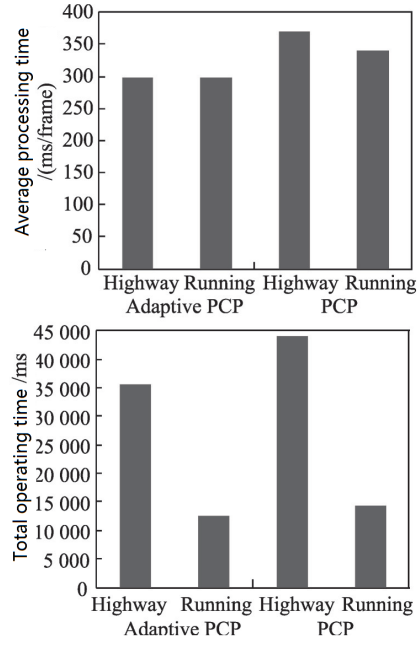


Figure 3. Comparison of running time between PCP and Adaptive PCP

Figure 4. Detection result of 740$^{\text{th}}$ frame of this paper method

the method in detecting and tracking of moving objects. Comparison of results generated by the proposed method with those of other methods showed that more reliable results could be obtained using the proposed method in real-time.

Further, the experimental data is taken from the VOT2014 dataset (`https://www.votchallenge.net/vot2014/dataset.html`) [49]. In order to ensure the effectiveness and stability of the algorithm, the moving target is selected here, where the dataset name is jogging. Detect and track the moving object with the largest moving area in the first 22 frames of video of each data set. This video is a video of the movement of two joggers. The feature is that the background is relatively single,

Figure 5. Various targets are detected in the detection algorithm

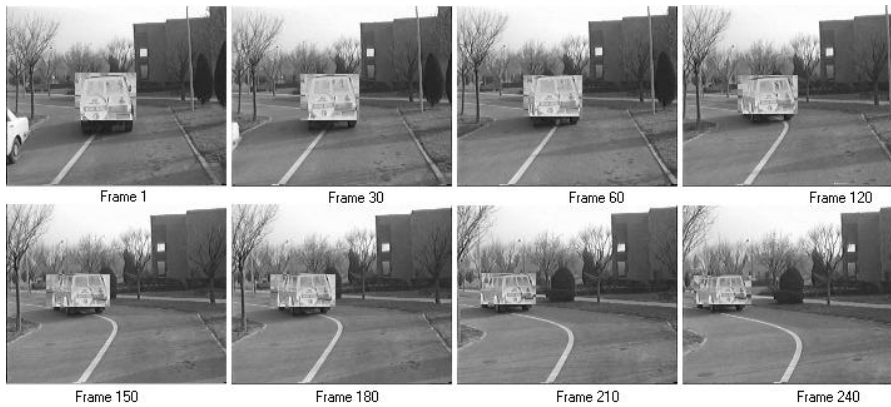
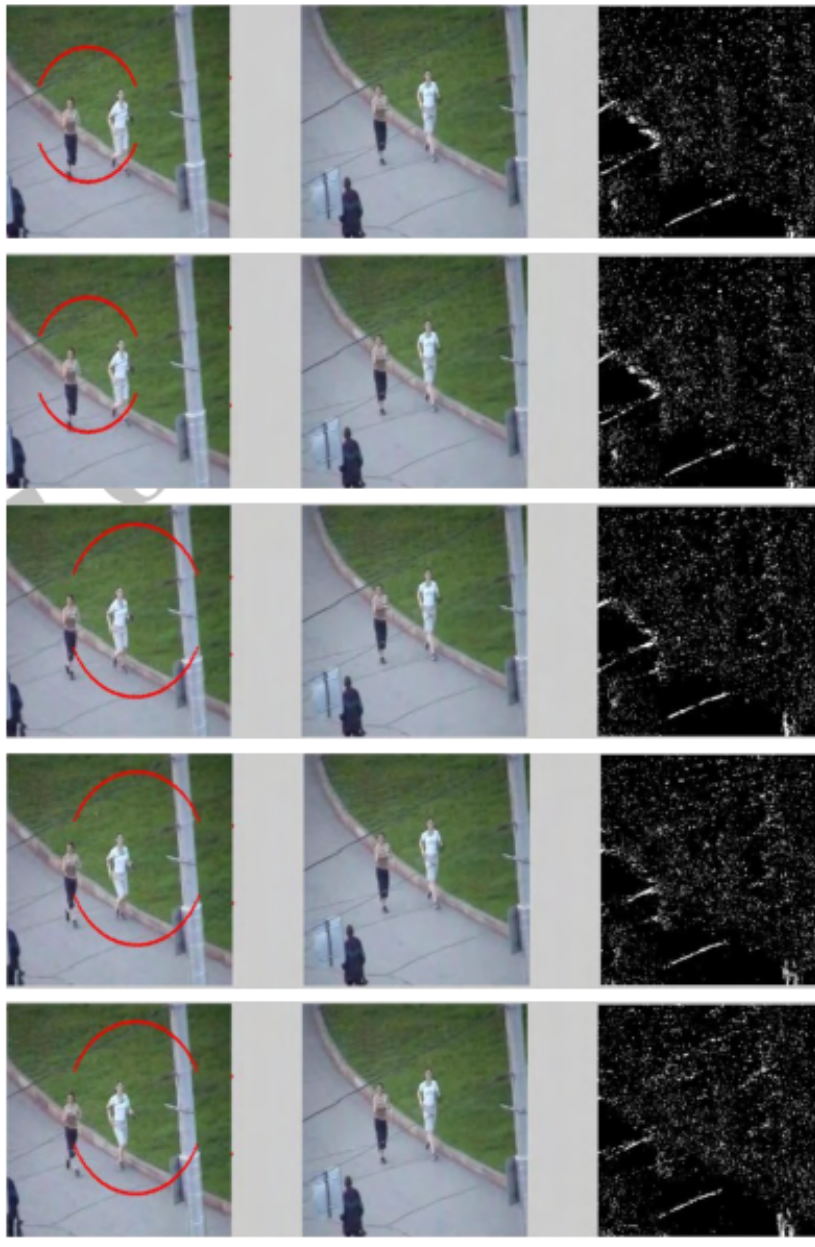


Figure 6. Tracking a vehicle while it turns over and changes its size and shape

but the two targets appear too close to each other. At this time, they are treated as a moving target for tracking and detection. The first five frames are shown in Figure 7. The experimental results show that the background of the entire video is relatively stable. When using the algorithm in this article to track two joggers, the target is almost marked in the ellipse.

6 CONCLUSIONS

In this paper, an adaptive low-rank sparse decomposition algorithm is proposed or it is known as adaptive principal component pursuit (A-PCP). First, a part of video frames are loaded, and the initial background A_b is obtained by low-rank sparse decomposition, and singular value decomposition is performed to obtain a low-dimensional subspace A_{sub}, A_{sub} connects with a new number of frames of the calculated video background vector, an augmented low-rank matrix is constructed. Then, a low-rank sparse decomposition of the objective function containing $\hat{A}$ results in a new low-rank background matrix A. Finally, the low-dimensional background



a) Current frame image

b) Background extracted by the algorithm in this paper

c) Sports targets of interest

Figure 7. Jogging target tracking effect map

is updated with the current A model A_{sub}, the above process is repeated until all frames are processed.

This adaptive low-rank sparse decomposition algorithm is applied to the detection of moving targets in intelligent video surveillance. In the algorithm, the background model A_b is established by low-rank sparse decomposition, and it is reduced to the low-dimensional space A_{sub}. The background of the new frame is obtained by performing low-rank sparse decomposition on the augmented matrix containing A_{sub}, and then the current background vector is used to update the background model. In this paper, an adaptive process is introduced in the traditional PCP algorithm, experiments show that the robustness of the algorithm is improved to complex backgrounds, and it can achieve better detection results. At the same time, compared to the PCP algorithm, all frame vectors are loaded at once, this adaptive update mechanism can improve the robustness of the algorithm. The mechanism of partially loading video frames can also reduce the delay of the algorithm and avoid memory overflow at the same time, thus the overall performance of moving target recognition is improved.

Acknowledgements

The project is supported by the Scientific Research Fund of Hunan Provincial Education Department (No. 16C0804, Research on Personalized Push Algorithm of Unstructured Education Resources Based on Mobile Micro-Lectures), China.

REFERENCES

[1] Garcia-Garcia, B.—Bouwmans, T.—Rosales Silva, A. J.: Background Subtraction in Real Applications: Challenges, Current Models and Future Directions. Computer Science Review, Vol. 35, 2020, Art. No. 100204, doi: 10.1016/j.cosrev.2019.100204.

[2] Babaee, M.—Dinh, D. T.—Rigoll, G.: A Deep Convolutional Neural Network for Video Sequence Background Subtraction. Pattern Recognition, Vol. 76, 2018, pp. 635–649, doi: 10.1016/j.patcog.2017.09.040.

[3] Lim, L. A.—Keles, H. Y.: Foreground Segmentation Using Convolutional Neural Network for Multiscale Feature Encoding. Pattern Recognition Letters, Vol. 112, 2018, pp. 256–262, doi: 10.1016/j.patrec.2018.08.002.

[4] Choo, S.—Seo, W.—Jeong, D.—Cho, N. I.: Multi-Scale Recurrent Encoder-Decoder Network for Dense Temporal Classification. 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 2018, pp. 103–108, doi: 10.1109/icpr.2018.8545597.

[5] Choo, S.—Seo, W.—Jeong, D.—Cho, N. I.: Learning Background Subtraction by Video Synthesis and Multi-Scale Recurrent Networks. In: Jawahar, C., Li, H., Mori, G., Schindler, K. (Eds.): Computer Vision – ACCV 2018. Springer, Cham,

Lecture Notes in Computer Science, Vol. 11366, 2018, pp. 357–372, doi: 10.1007/978-3-030-20876-9_23.

[6] Gracewell, J.—John, M.: Dynamic Background Modeling Using Deep Learning Autoencoder Network. Multimedia Tools and Applications, Vol. 79, 2020, pp. 4639–4659, doi: 10.1007/s11042-019-7411-0.

[7] Yang, Y.—Zhang, T.—Hu, J.—Xu, D.—Xie, G.: End to End Background Subtraction via a Multi-Scale Spatio-Temporal Model. IEEE Access, Vol. 7, 2019, pp. 97949–97958, doi: 10.1109/access.2019.2930319.

[8] Qiu, M.—Li, X.: A Fully Convolutional Encoder-Decoder Spatial-Temporal Network for Real-Time Background Subtraction. IEEE Access, Vol. 7, 2019, pp. 85949–85958, doi: 10.1109/access.2019.2925913.

[9] Minematsu, T.—Shimada, A.—Uchiyama, H.—Taniguchi, R.: Analytics of Deep Neural Network-Based Background Subtraction. MDPI Journal of Imaging, Vol. 4, 2018, No. 6, Art. No. 78, 19 pp., doi: 10.3390/jimaging4060078.

[10] García-González, J.—Ortiz-de-Lazcano-Lobato, J. M.—Luque-Baena, R. M.—López-Rubio, E.: Background Modeling by Shifted Tilings of Stacked Denoising Autoencoders. In: Ferrández Vicente, J., Álvarez-Sánchez, J., de la Paz López, F., Toledo Moreo, J., Adeli, H. (Eds.): From Bioinspired Systems and Biomedical Applications to Machine Learning (IWINAC 2019). Springer, Cham, Lecture Notes in Computer Science, Vol. 11487, 2019, pp. 307–316, doi: 10.1007/978-3-030-19651-6_30.

[11] García-González, J.—Ortiz-de-Lazcano-Lobato, J. M.—Luque-Baena, R. M.—Molina-Cabello, M. A.—López-Rubio, E.: Foreground Detection by Probabilistic Modeling of the Features Discovered by Stacked Denoising Autoencoders in Noisy Video Sequences. Pattern Recognition Letters, Vol. 125, 2019, pp. 481–487, doi: 10.1016/j.patrec.2019.06.006.

[12] Stauffer, C.—Grimson, W. E. L.: Adaptive Background Mixture Models for Real-Time Tracking. Proceedings of the 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Fort Collins, USA, Vol. 2, 1999, pp. 246–252, doi: 10.1109/cvpr.1999.784637.

[13] Elgammal, A. M.—Harwood, D.—Davis, L. S.: Nonparametric Model for Background Subtraction. In: Vernon, D. (Ed.): Computer Vision – ECCV 2000. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1843, 2010, pp. 751–767, doi: 10.1007/3-540-45053-x_48.

[14] Kim, K.—Chalidabhongse, T.—Harwood, D.—Davis, L.: Real-Time Foreground-Background Segmentation Using Codebook Model. Real-Time Imaging, Vol. 11, 2005, No. 3, pp. 172–185, doi: 10.1016/j.rti.2004.12.004.

[15] Maddalena, L.—Petrosino, A.: A Self-Organizing Approach to Background Subtraction for Visual Surveillance Applications. IEEE Transactions on Image Processing, Vol. 17, 2008, No. 7, pp. 1168–1177, doi: 10.1109/tip.2008.924285.

[16] Maddalena, L.—Petrosino, A.: The SOBS Algorithm: What Are the Limits? Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, Providence, USA, 2012, pp. 21–26, doi: 10.1109/CVPRW.2012.6238922.

[17] Barnich, O.—Van Droogenbroeck, M.: ViBe: A Universal Background Subtraction Algorithm for Video Sequences. IEEE Transactions on Image Processing, Vol. 20, 2011, No. 6, pp. 1709–1724, doi: 10.1109/tip.2010.2101613.

[18] Hofmann, M.—Tiefenbacher, P.—Rigoll, G.: Background Segmentation with Feedback: The Pixel-Based Adaptive Segmenter. Proceedings of the 2012 IEEE Computer Society Conference on Computer Vision Pattern Recognition Workshops, 2012, pp. 38–43, doi: 10.1109/cvprw.2012.6238925.

[19] Donoho, D. L.: Compressed Sensing. IEEE Transactions on Information Theory, Vol. 52, 2006, No. 4, pp. 1289–1306, doi: 10.1109/tit.2006.871582.

[20] Candès, E. J.—Romberg, J. K.—Tao, T.: Stable Signal Recovery from Incomplete and Inaccurate Measurements. Communications on Pure and Applied Mathematics, Vol. 59, 2006, No. 8, pp. 1207–1223, doi: 10.1002/cpa.20124.

[21] Vaswani, N.—Zahzah, E.: Robust PCA and Robust Subspace Tracking. Preprint, Arxiv 2017.

[22] Bouwmans, T. et al.: Robust PCA via Principal Component Pursuit: A Review for a Comparative Evaluation in Video Surveillance. Special Issue on Background Models Challenge, Computer Vision and Image Understanding (CVIU), Vol. 122, 2014, No. 2, pp. 22–34, doi: 10.1016/j.cviu.2013.11.009.

[23] Candès, E. J.—Li, X. D.—Ma, Y.—Wright, J.: Robust Principal Component Analysis? Journal of the ACM (JACM), Vol. 58, 2011, No. 3, Art. No. 11, 37 pp., doi: 10.1145/1970392.1970395.

[24] Zhou, X. W.—Yang, C.—Yu, W. C.: Moving Object Detection by Detecting Contiguous Outliers in the Low-Rank Representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 35, 2013, No. 3, pp. 597–610, doi: 10.1109/tpami.2012.132.

[25] Gao, Z.—Cheong, L. F.—Wang, Y. X.: Block-Sparse RPCA for Salient Motion Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 36, 2014, No. 10, pp. 1975–1987, doi: 10.1109/tpami.2014.2314663.

[26] Liu, X.—Zhao G. Y.—Yao J. W.—Qi, C.: Background Subtraction Based on Low-Rank and Structured Sparse Decomposition. IEEE Transactions on Image Processing, Vol. 24, 2015, No. 8, pp. 2502–2514, doi: 10.1109/tip.2015.2419084.

[27] Bouwmans, T.—Sobral, A.—Javed, S.—Jung, S. K.—Zahzah, E.: Decomposition into Low-Rank Plus Additive Matrices for Background/Foreground Separation: A Review for a Comparative Evaluation with a Large-Scale Dataset. Computer Science Review, Vol. 23, 2017, pp. 1–71, doi: 10.1016/j.cosrev.2016.11.001.

[28] Jolliffe, I. T.: Graphical Representation of Data Using Principal Components. Chapter 5. In: Jolliffe, I. T.: Principal Component Analysis. Springer, New York, Springer Series in Statistics, 2010, pp. 41–64, doi: 10.1007/978-1-4757-1904-8_5.

[29] Wright, J.—Peng, Y.—Ma, Y.—Ganesh, A.—Rao, S.: Robust Principal Component Analysis: Exact Recovery of Corrupted Low-Rank Matrices by Convex Optimization. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., Culotta, A. (Eds.): Advances in Neural Information Processing Systems 22 (NIPS 2009). 2009, 9 pp.

[30] Becker, S.—Candes, E.—Grant, M.: TFOCS: Flexible First-Order Methods for Rank Minimization. Low-Rank Matrix Optimization Symposium, SIAM Conference on Optimization, 2011.

[31] Tang, G.—Nehorai, A.: Robust Principal Component Analysis Based on Low-Rank and Block-Sparse Matrix Decomposition. 2011 45th Annual Conference on Information Sciences and Systems (CISS 2011), Baltimore, MD, USA, 2011, doi: 10.1109/ciss.2011.5766144.

[32] Wohlberg, B.—Chartrand, R.—Theiler, J.: Local Principal Component Pursuit for Nonlinear Datasets. International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2012), Kyoto, Japan, 20102, pp. 3925–3928, doi: 10.1109/icassp.2012.6288776.

[33] Lin, Z.—Chen, M.—Wu, L.—Ma, Y.: The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. arXiv preprint arXiv:1009.5055, 2010.

[34] Yuan, X.—Yang, J.: Sparse and Low-Rank Matrix Decomposition via Alternating Direction Methods. Pacific Journal of Optimization, Vol. 9, 2009, No. 1, pp. 1–11.

[35] Rodríguez, P.—Wohlberg, B.: Fast Principal Component Pursuit Via Alternating Minimization. IEEE International Conference on Image Processing (ICIP 2013), Melbourne, Australia, 2013, pp. 69–73, doi: 10.1109/icip.2013.6738015.

[36] Guyon, C.—Bouwmans, T.—Zahzah, E.: Moving Object Detection via Robust Low Rank Matrix Decomposition with IRLS Scheme. In: Bebis, G. et al. (Eds.): Advances in Visual Computing (ISVC 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7431, 2012, pp. 665–674, doi: 10.1007/978-3-642-33179-4_63.

[37] Qiu, C.—Vaswani, N.: Real-Time Robust Principal Components' Pursuit. 2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2010, pp. 591–598, doi: 10.1109/allerton.2010.5706961.

[38] Narayanamurthy, P.—Vaswani, N.: MEDRoP: Memory-Efficient Dynamic Robust PCA. Preprint, December 2017, arXiv:1712.06061v1.

[39] Rodriguez, P.—Wohlberg, B.: Incremental Principal Component Pursuit for Video Background Modeling. Journal of Mathematical Imaging and Vision, Vol. 55, 2016, No. 1, pp. 1–18, doi: 10.1007/s10851-015-0610-z.

[40] Javed, S.—Mahmood, A.—Bouwmans, T.—Jung, S. K.: Background-Foreground Modeling Based on Spatiotemporal Sparse Subspace Clustering. IEEE Transactions on Image Processing, Vol. 26, 2017, No. 12, pp. 5840–5854, doi: 10.1109/tip.2017.2746268.

[41] Sobral, A.—Bouwmans, T.—Zahzah, E.: Double-Constrained RPCA Based on Saliency Maps for Foreground Detection in Automated Maritime Surveillance. ISBC 2015 Workshop conjunction with AVSS 2015, Karlsruhe, Germany, 2015, pp. 1–6, doi: 10.1109/avss.2015.7301753.

[42] Javed, S.—Bouwmans, T.—Jung, S. K.: Combining ARF and OR-PCA Background Subtraction of Noisy Videos. In: Murino, V., Puppo, E. (Eds.): Image Analysis and Applications – ICIAP 2015. Springer, Cham, Lecture Notes in Computer Science, Vol. 9280, 2015, pp. 340–351, doi: 10.1007/978-3-319-23234-8_32.

[43] Zhou, P.—Feng, J.: Outlier-Robust Tensor PCA. 2017 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA, 2017, pp. 3938–3946, doi: 10.1109/cvpr.2017.419.

[44] Hu, W.—Yang, Y.—Zhang, W.—Xie, Y.: Moving Object Detection Using Tensor Based Low-Rank and Saliently Fused-Sparse Decomposition. IEEE Transactions on Image Processing, Vol. 26, 2017, No. 2, pp. 724–737, doi: 10.1109/tip.2016.2627803.

[45] Sobral, A.—Javed, S.—Jung, S. K.—Bouwmans, T.—Zahzah, E.: Online Stochastic Tensor Decomposition for Background Subtraction in Multispectral Video Sequences. 2015 IEEE International Conference on Computer Vision Workshop (ICCVW), Santiago, Chile, 2015, pp. 946–953, doi: 10.1109/iccvw.2015.125.

[46] Javed, S.—Bouwmans, T.—Jung, S. K.: Stochastic Decomposition into Low Rank and Sparse Tensor for Robust Background Subtraction. 6th International Conference on Imaging for Crime Prevention and Detection (ICDP 2015), 2015, doi: 10.1049/ic.2015.0105.

[47] Yang, F.—Jiang, H.—Shen Z. W.—Deng, W.—Metaxas, D.: Adaptive Low Rank and Sparse Decomposition of Video Using Compressive Sensing. Proceedings of the 2013 IEEE International Conference on Image Processing, Melbourne, Australia, 2013, pp. 1016–1020, doi: 10.1109/icip.2013.6738210.

[48] Brand, M.: Fast Low-Rank Modifications of the Thin Singular Value Decomposition. Linear Algebra and Its Applications, Vol. 415, 2006, No. 1, pp. 20–30, doi: 10.1016/j.laa.2005.07.021.

[49] Li, W.—Li, X.: Multiple Object Tracking Based on Modified Algorithm of GMMCP Tracker. IEEE International Conference on Signal and Image Processing (ICSIP), 2017, pp. 11–15, doi: 10.1109/siprocess.2016.7888214.

Jiang Chong received her Bachelor's degree in computer science and technology from Hunan University in 2002. Then she obtained her Master's degree and her Ph.D. in computer application technology from Central South University in Changsha, China. Now she is Researcher at the School of Computer Science and Engineering, Hunan Women's University, China. Her research interests include machine learning, digital image and video processing directions.

Computing and Informatics, Vol. 39, 2020, 1082–1098, doi: 10.31577/cai_2020_5_1082

IMPROVED DEEP FOREST MODE FOR DETECTION OF FRAUDULENT ONLINE TRANSACTION

Mian Huang

The Key Laboratory of Embedded System and Service Computing of Ministry of Education
Tongji University
Shanghai, China
e-mail: net-cn@163.com

Lizhi Wang, Zhaohui Zhang*

School of Computer Science and Technology
Donghua University
Shanghai, China
e-mail: lizhi_wang2769433@163.com, zhzhang@dhu.edu.cn

Abstract. As the rapid development of online transactions, transaction frauds have also emerged seriously. The fraud strategies are characterized by specialization, industrialization, concealment and scenes. Anti-fraud technologies face many challenges under the trend of new situations. In this paper, aiming at sample imbalance and strong concealment of online transactions, we enhance the original deep forest framework to propose a deep forest-based online transaction fraud detection model. Based on the BaggingBalance method we propose, we establish a global sample imbalance processing mechanism to deal with the problem of sample imbalance. In addition, the autoencoder model is introduced into the detection model to enhance the representation learning ability. Via the three-month real online transactions data of a China's bank, the experimental results show that, evaluating by the metric of precision and recall rate, the proposed model has a beyond 10 % improvement compared to the random forest model, and a beyond 5 % improvement compared to the original deep forest model.

Keywords: Deep forest, online transaction, fraud detection, autoencoder

* Corresponding author

1 INTRODUCTION

Under the general trend of Internet finance, digital technologies such as big data and artificial intelligence (AI) are widely used in the financial field, and the volume and potential development of financial markets are gradually enlarged. At the same time, the risk of exposure is also increasing, and frauds are endless [1]. According to statistics [2], China's fraudulent employees exceed 1.5 million, and the raised annual output value reaches 100 billion in 2017. The financial institutions that use Internet financial technology to carry out financial business are one of the main targets of the attack. The risk control of digital finance faces enormous challenges.

At the background, detecting fraudulent transaction patterns precisely is a highly important research direction in the field of online transaction fraud detection. The traditional expert rule-driven fraud detection technologies require a lot of manual operations, have a high application cost and low efficiency, while the traditional anti-fraud technologies consider simple transaction dimensions, thus they are difficult to form a multi-dimensional user portrait for the user. The online transactions have strong real-time performance, large amount of data, and fraud is characterized by small amount and high frequency. It is challenging for traditional anti-fraud methods to precisely detect fraudulent online transactions.

At present, a large number of machine learning (ML) – based research are widely used in the field of fraud detection, including decision trees [3], support vector machines (SVM) [3], naive bayes [4], random forest (RF) [4,5] and other ML algorithms. ML technology learns existing fraud strategies and explores potential fraud strategies by learning historical transaction information for online transactions, then precisely detects online transactions with fraudulent possibilities. In addition, some research about deep learning (DL) techniques are gradually being used in fraud detection tasks.

DL techniques [6] such as convolutional neural networks (CNN) and recurrent neural networks (RNN) have achieved excellent performance in many popular tasks, such as image recognition [7], natural language processing [8], and so on. DL techniques excel in processing high-dimensional data and nonlinear feature space inputs, which are common in fraud detection tasks. On this basis, some studies begin to introduce DL for fraud detection, and use its powerful representation learning ability to solve the problem of online transaction fraud detection. A research from McKinsey concluded that it is a promising solution to apply DL techniques for the problem of financial fraud detection [9]. However, using only ML techniques or DL techniques does not completely solve the problem of fraud detection [10]. Therefore, integrating the advantages of ML and DL for fraud detection tasks has also become one of the research directions.

Deep forest (multi-Grained Cascade Forest, gcForest) is a novel decision tree ensemble method, which may open the door towards an alternative to deep neural networks for many tasks [11]. By creating a cascade forest structure, the method could enable its representation learning. At the same time, its multi-scanning structure could enhance its representational learning ability. From another perspective,

gcForest is a learning framework that integrates ML and DL techniques. The multi-scanning structure uses the idea of 1D and 2D convolution similar to CNN to establish representation learning. Based on the idea of stacking, cascade forest structure ensembles the RF model [12] and the completely random forest model [13] as base classifiers.

In this paper, we propose an improved gcForest-based online transaction fraud detection model. In view of the problems in online transaction fraud detection, on the one hand, we add the autoencoder DL model [7] to the multi-scanning structure to enhance its representation learning ability, because autoencoder could produce more concise unsupervised representations, which is proved to be a robust algorithm [14]. At the same time, we use XGBoost (eXtreme Gradient Boosting) model [15] to replace the completely random forest base classifiers in cascade forest structure. By combining with the proposed BaggingBalance method, a global sample imbalance processing mechanism is established. XGBoost is a scalable end-to-end tree boosting system [15], which is used widely to achieve state-of-art results on many ML competitions [16]. By combining the above methods, we enhance the original gcForest framework, then establish a detection model for online transaction fraud. The main contributions of this paper are summarized as follows:

- Apply gcForest model and improve the model for fraud detection in online transactions. Based on the accumulated experience of ML in fraud detection tasks in recent years, and with the excellent representation learning ability demonstrated by DL, the structure of the original gcForest is improved for the online transaction fraud detection task, and the experiment result shows the proposed model is superior to RF model and the original gcForest model.
- Aiming at the data characteristics of online transactions, the multi-scanning structure of the original gcForest is enhanced. Autoencoder model with excellent representation learning ability is introduced to enhance the model's feature learning of online transactions.
- The BaggingBalance method is proposed to deal with the sample imbalance problem in online transactions on the data input. At the same time, the XGBoost model is introduced in the cascade forest structure. Combined with the two, a global sample imbalance processing mechanism is established.

In the remainder of the paper, Section 2 describes some related work about status quo of the online transaction fraud detection. Section 3 introduces the methodology proposed in this paper. The data information and experimental results are discussed in Section 4. Finally, conclusion and future work are presented.

2 RELATED WORK

Nowadays, with the continuous development of Internet finance, online transaction fraud detection has become a hot research topic, including credit card fraud de-

tection, mobile payment fraud detection, B2C (Business-to-Customer) transaction fraud detection and so on.

The ML-based fraud detection algorithm is widely used in the field of online transaction fraud detection, including supervised learning model and unsupervised learning model. The supervised learning models establish a fraud detection model based on historical transaction data after manual investigation to determine whether a new transaction is fraudulent. For example, Shiyang Xuan et al. [5] learn the behavior patterns of normal and abnormal transactions via two kinds of RFs, where the two RFs have different base classifiers, and evaluate their performance on credit card transactions. While unsupervised learning models typically treat identified outliers as detected fraudulent transactions using outlier detection or anomaly detection techniques. In 2014, Olszewski [17] uses the self-organizing map (SOM) method to build a user behavior model to look for outliers that deviate from normal user behavior for fraud detection. ML-based detection algorithms have the advantages of learning known fraud patterns and detecting potential new fraud strategies. However, the methods of supervised learning strongly rely on the correctness of the original labels and the need to deal with the existing sample imbalance. Unsupervised learning is very sensitive to the overlapping distribution of normal transactions and fraudulent transactions, which often leads to a serious decline in accuracy [18].

With the excellent performance of DL technology in many classification tasks, DL technology is introduced in the field of online transaction fraud detection. In 2016, Kang Fu et al. [19] propose a CNN-based fraud detection framework, which could learn fraud behavior patterns via transaction data and show its excellent performance compared with some state-of-art methods. In 2017, Jingdong Finance's Shuhao Wang et al. [20] present CLUE framework, a novel DL-based transaction fraud detection system. By using neural network based embedding and RNN, the system achieves over 3 times improvement over the existing fraud detection approaches on real production data for eight months. In 2018, Zhaohui Zhang et al. [21] apply CNN for the task of online transaction fraud detection by constructing an input feature sequencing layer to obtain various input feature patterns, the proposed method outperforms the existing CNN model. At the same year, Abhimanyu Roy et al. [22] deeply study the application of DL technologies in credit card fraud detection tasks, and solve the common problems in fraud by using high-performance distributed cloud computing environment, while providing a parameter adjustment framework for DL topology. However, although DL technology can acquire more sequential information between transactions, it is insufficient for DL to just learn feature information within a single transaction, which can be well learned by ML technologies. But only using ML methods would attenuate the sequential learning ability of detection models [10].

In recent years, the online transaction fraud detection field starts to apply detection techniques that combine the advantages of ML and DL. In 2017, Xurui Li et al. [10] propose a novel "within-between-within" (WBW) sandwich-structured sequence learning architecture by integrating ensemble and DL methods, and intro-

duce attention mechanism to further enhance its performance. In the same year, Zahra Kazemi and Houman Zarrabi [23] use deep autoencoder model and softmax network to learn credit card transaction information and establish a fraud detection model, where results show the advantages of proposed method comparing to state-of-art methods.

In this work, based on the framework of the original gcForest, we improve the model for the online transaction fraud detection task. Introducing the autoencoder model into the multi-scanning structure enables the detection model a stronger representation learning ability on the input of the cascade forest, which could better handle the strong concealment of online transaction fraud patterns. While establishing a global sample imbalance processing mechanism, which could deal with the problem of sample imbalance in online transaction fraud detection. On this basis, we propose an improved gcForest-based method for online transaction fraud detection.

3 METHODOLOGY

3.1 Improved gcForest Framework for Detecting Fraudulent Online Transaction

The improved gcForest-based online transaction fraud detection framework can be seen in Figure 1, including the multi-scanning and the cascade forest. The cascade forest structure uses XGBoost as the base classifier to replace the completely random forest model in the original gcForest. As for the multiscanning structure, it introduces the autoencoder model into the original structure to enhance representational learning, then reconstruct a multi-scanning structure based on autoencoder combined with sample imbalance processing method BaggingBalance.

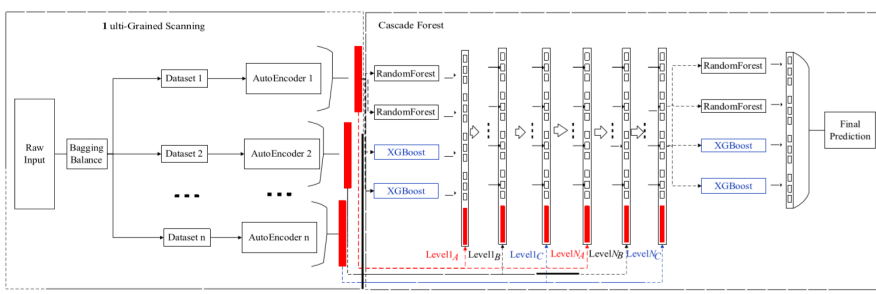


Figure 1. Improved gcForest framework for detecting fraudulent online transaction

As shown in Figure 1, assuming that the features number of initial input is M, there are n autoencoders for multi-scanning structure. For training samples with size N, an autoencoder obtains its hidden layer output representation vector H through the BaggingBalance method. the vector H will be used to train the first level of the first layer of the cascade forest. The same operation is performed on

the other $n-1$ autoencoders, and the obtained hidden layer output vectors are respectively used to train the second level to k^{th} level of the first layer of the cascade forest.

Repeat the same operations for every initial training transaction sample. The expanded feature vector adds the class vectors generated by the previous level, which are used to train the second and third layers of the cascaded forest, respectively, and this process is repeated until the convergence of the model performance. In other words, the final model is actually a ensemble of deep forest, each of which is composed of multiple levels, as shown in Figure 1, with each layer corresponding to a hidden layer vector representation of an autoencoder.

3.2 BaggingBalance: A Method for Processing Sample Imbalance

BaggingBalance is a sample imbalance processing method based on the idea of Bagging [24], by under-sampling operation of raw data at the data input layer, and randomly selecting attribute features, thus obtaining different sampling data sets.

Specifically, the original data set is first divided into a majority class training set D_{major} and a minority class training set D_{minor} based on bootstraping [24]. Sampling the majority class training sets produces a data set D_{sample}: each time randomly pick a sample from the majority class dataset D_{major}, copy it into D_{sample}, and then put the sample back into the initial dataset D_{major}. It is possible to enable the sample sampled at the next sampling via the step. Different from self-sampling, the times of this process is repeatedly executed is the sample size $|\mathrm{D}_{minor}|$ of the minority training set D_{minor}, instead of the size of $|\mathrm{D}_{major}|$ of majority class training set D_{major}.

In addition, unlike Bagging, which only differs by sample perturbation, the BaggingBalance method also introduces the randomness of attribute features, which is similar to the idea of RF, i.e., the attribute feature perturbation is added at the same time, which will improve the generalization performance of final model.

The process of BaggingBalance algorithm is Algorithm 1.

Algorithm 1 BaggingBalance

Input: The majority training set, D_{major}, the minority training set, D_{minor}, the feature space, F, the number of sampling training set, k, and number of features randomly selected, $\mathrm{m}_{feature}$

Output: k sample training sets, $\mathrm{D} = \{D_1, D_2, \ldots, D_k\}$;

$\mathrm{D} = []$;

for $i = 0$ to k **do**

Sampling D_{major} to get sampled data set D_{sample}, where $|\mathrm{D}_{sample}| = |\mathrm{D}_{minor}|$;

Randomly extract feature subset F_{sample} from feature space F, where $|\mathrm{F}_{sample}| = \mathrm{m}_{feature}$;

$D_i = \{\mathrm{D}_{sample}, \mathrm{F}_{sample}\}$

end forreturn D;

3.3 New Multi-Scanning Structure Using Autoencoder

In the original gcForest algorithm, the multi-scanning structure [11] uses the sliding window technique to process the original features. The vectors obtained by each sliding are processed by the RF model and completely random forest model to obtain the class vector, and then all the class vectors are concatenated as a transformation feature vector, which is passed as an input feature vector to the cascade forest for classification.

However, the sliding window-based method has its own limitations. As mentioned in the proposed paper [11], the multi-scanning structure has a good effect on data with sequence relationship or spatial relationship. Because the sliding window method can only slide linearly, thus there is a great demand for the feature space arrangement of the raw data. Specifically, the sliding window is qualified to process the feature vectors with sequence relationship, but there is no strong sequence relationship between the original feature vectors in each online transaction, even in an out-of-order feature space status. In addition, the reconstructed feature vector generated by the original structure is completely composed of the class vectors, which cannot fully map the feature space of the original data.

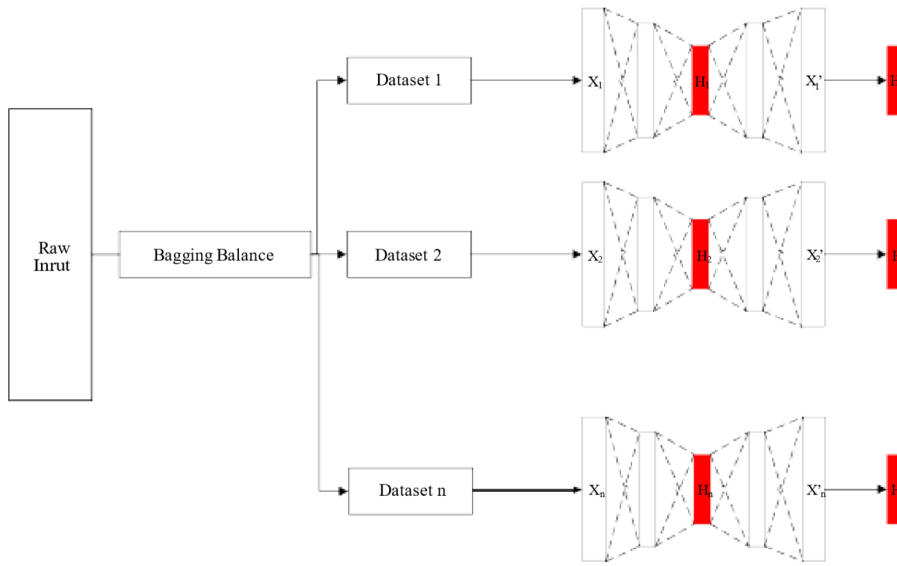


Figure 2. New multi-scanning structure using autoencoder

Therefore we reconstruct the multi-scanning structure by introducing autoencoder algorithm and the BaggingBalance method, and propose a new multi-scanning structure using autoencoder, which is shown in Figure 2.

On the one hand, based on the BaggingBalance method we proposed, the original input data is double-randomly sampled in the sample space and the feature

space, and several different sampling training sets are obtained. Through the introduction of randomness, the disorder of the online transaction feature space is considered while dealing with the problem of sample imbalance. Through the random feature selection, the random combination of different transaction attributes in the original feature space can be realized, and the internal relationship between the fraud patterns and the transaction feature space in online transaction can be deeply explored. This is also reason that why randomness exists in many ML algorithms.

On the other hand, the autoencoder model is introduced in the multi-scanning structure to further enhance the representation learning ability of the fraud detection model. Autoencoder has proven to be a robust algorithm which can be used in several applications and the main advantage is to extract best features for data analysis [23]. The sampling training set obtained by the BaggingBalance method is used as input to train the autoencoder model, and the hidden layer output representation vector of the trained autoencoder is extracted as a new modified feature vector, which is transmitted as input to the cascade forest model for model training. Compared with the class vector generated by RF model and completely random forest model, the hidden layer output representation vector obtained by the autoencoder is a better expression of the original input feature space. What is more, it is more concise and effective, and more fully reflects the distribution of the original feature space.

The overall process flow of the multi-scanning structure using autoencoder is summarized as shown in Algorithm 2.

Algorithm 2 The process flow of the multi-scanning structure using autoencoder

Input: The majority training set, D_{major}, the minority training set, D_{minor}, the feature space, F, the number of initialized autoencoders, k, the number of features randomly selected, $\mathrm{m}_{feature}$, and the number of iterations of the autoencoder, iters

Output: output expression vector of k autoencoders in hidden layer, $\mathrm{H} = \{H_1, H_2, \ldots, H_k\}$;

```
H = [];
for each AutoEncoder_i in k autoencoders do
    Sampling D_major to get sampled data set D_sample, where |D_sample| = |D_minor|;
    Randomly extract feature subset F_sample from feature space F, where |F_sample| = m_feature;
    for t = 0 to iters do
        training AutoEncoder_i by TrainAutoencoder(D_major, D_minor, F_sample);
    end for
    get H_i;
    push H_i to H;
end forreturn H;
```

4 EXPERIMENTS

4.1 Datasets and Indicators

Experimental data comes from real online transaction data of a China's bank, including three-month B2C transaction records (from April 2017 to June 2017). There are original 67 available transaction attributes, and there are more than 70 000 transactions labeled as fraudulent transactions in historical data. In this paper, we use transaction data of the first two months as a training set to train the improved gcForest-based online transaction fraud detection model. The last months transaction data is used as the testing set to evaluate the performance of the detection model. Last but not least, precision rate and recall rate is used to evaluate the performance of the proposed model.

Real / Predicted	True Fraud	True Normal
Predicted Fraud	TP	FP
Predicted Normal	FN	TN

Table 1. Confusion matrix

As shown in Table 1, because it is a fraudulent transaction interception, the focus of the model should be on fraudulent transactions, so the confusion matrix is slightly modified. TP (True Positive) is the number of fraudulent transactions judged as fraudulent transactions by the model. FP (False Positive) is the number of normal transactions that are judged as fraudulent transactions by the model. TN (True Negative) is the number of normal transactions that are judged as normal transactions by the model. FN (False Negative) is the number of fraudulent transactions that are judged as normal transactions by the model. Then, 3 indicators in Table 2 will serve to evaluate the performance.

Indicator Name	Calculation Method
Accuracy	$(TP + TN)/(TP + TN + FP + FN)$
Precision	$TP/(TP + FP)$
Recall	$TP/(TP + FN)$

Table 2. Indicator calculation method

As shown in Table 2, in fraud detection, the accuracy rate refers to the ratio of the number of correct transactions predicted by the detection model to the total electronic transactions. Accuracy is the most commonly used performance metric in classification tasks, which is suitable for both binary classification tasks and multi-classification tasks. However, it cannot meet the needs of fraud detection tasks because this indicator cannot accurately measure the performance of fraud detection models due to the imbalance of samples in fraud detection. On this basis, the concepts of precision and recall are proposed, matching with precision and recall

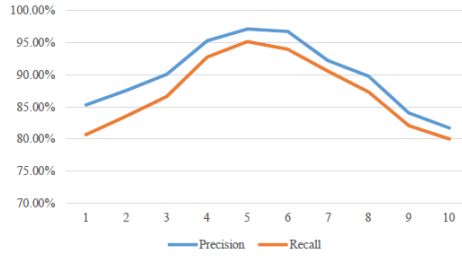
respectively in machine learning. Therefore, the application of precision rate and recall rate in fraud detection task can reflect the performance and effect of the model in electronic transaction fraud detection.

4.2 Model Evaluation

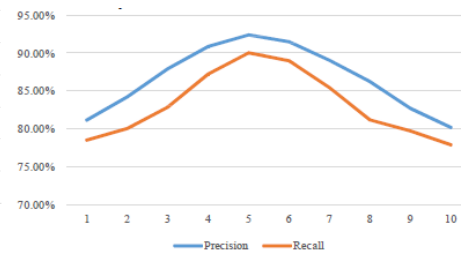
1. Selection of the Number of Autoencoder: In the framework of improved gcForest-based online transaction fraud detection, the selection of the autoencoders' number, i.e., the selection of the number of sample datasets in BaggingBalance, is a problem worth studying. Because the data distribution and feature distribution of sampling datasets generated by the BaggingBalance method tend to be very different due to the randomness of sample selection and feature selection. These datasets will be used as the input of the autoencoder model to train the model, and produce various hidden layer poor model performance. If there are too many autoencoders, overfitting will occur, which leads to the worse generalization ability of the model and the degraded performance.
2. Performance of the Fraud Detection Model: After determining the number of autoencoders, this section conducts an experimental study on the performance of the proposed fraud detection model. The RF model and the original gcForest model are selected as the comparison model. The test was extracted from the online transaction data of June 2017 which are divided into five subsets including the first 10 days, the first 15 days, the first 20 days, the first 25 days and the first 30 days.

Based on the above considerations, this section of the experiment selects the transaction data of the first two months as the training sets and tests it on the transaction data of the first 10 days, the first 20 days and the entire month in June. X-axis is the number of autoencoders which is considered in the ranges from 1 to 10. Figures 3 a), 3 b), 3 c) show the results of fraud detection models via the different autocoders' number, which are tested on the online transaction data for the first 10 days, the first 20 days and the entire month of June. From the above experimental results, the number of autoencoders should not be too large, and should not be too small, generally taking 4 to 6. If the number is too small, the dataset generated by BagingBalance is small in size and cannot fully reflect the original data space, resulting in insufficient learning of the original data and output representation, which has a great influence on the final detection effect of the model.

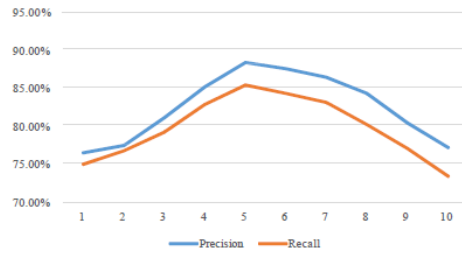
In addition, to verify the effectiveness of the introduced autoencoder in the model, we also evaluate the performance of the original cascaded forest structure with an autoencoder in this section. In this model, we input the raw online transaction data, pass them into the autoencoder and obtain the hidden layer output representation vector, which will be as the input of original cascaded forest structure. From Figures 4 a), 4 b) we can conclude that compared with the original gcForest model, the introduction of the autoencoder has its effectiveness. At the same time,



a) Data set of the first 10 days



b) Data set of the first 20 days

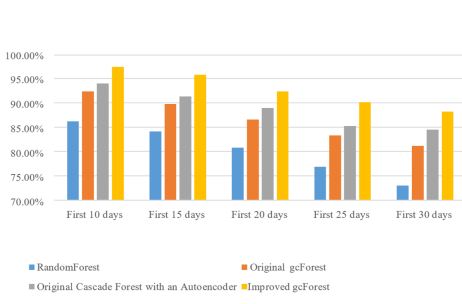


c) Data set of the whole month

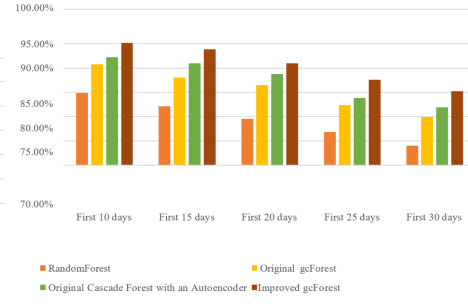
Figure 3. Evaluation on the number of autoencoders

the results also show that the improved gcForest with multi-autoencoders based on BaggingBalance is superior to ones with an autoencoder.

Based on the last experimental result, this section initializes the number of autoencoders to 5. Figure 4 a) shows the precision rate on different models in the five test sets, and Figure 4 b) shows the recall rate on different models in the five test sets. It can be seen that the proposed model has a beyond 10 % improvement compared to RF model, a beyond 5 % improvement compared to the original gcForest model.



a) The precision rate of different models



b) The recall rate of different models

Figure 4. The performances of different models on various sample sets

4.3 System Implementation

In order to verify the comprehensive performance of the model, a fraud detection subsystem was built. Based on the fraud detection model based on deep forest proposed in this paper, the system realized two functions of offline model training and simulated real-time electronic transaction fraud detection to verify the application and effectiveness of the detection model.

For the model off-line training module, the functions of data extraction, data preprocessing, feature differentiation and model training of the deep forest model are completed. The realization of this part of functions in order to make the training model process from data extraction to result analysis can be completed at the system end, convenient for users to control the training process of the model.

For the function of simulating real-time electronic transaction fraud detection module, the detection, interception and release of real-time transaction data are completed. The work of this part is to deploy the trained deep learning model into the system. The system passes the received electronic transaction into the detection model for detection. If the detection is normal, the transaction will be released. If the transaction is identified as fraudulent by the model, it will be intercepted.

After configuring the service based on the deep forest fraud detection model, the model will start running to prevent and monitor the risks of real-time electronic transaction data streams entering the system. The interactive page design for real-time risk control monitoring is shown in Figure 5. This part shows the real-time detection results after real-time transaction data enters the group behavior fraud detection subsystem and the performance analysis and visualization of the running detection model. The detection result display part displays basic information such as the user account of the current transaction, the user's name, the time of the transaction and the interception of the detection model. At the same time, the intercepted electronic transactions are displayed in detail to analyze the characteristics of the intercepted transactions, as shown in Figure 6.

At the same time, the simulated real-time electronic transaction fraud detection function counts the real-time detection performance indicators of the fraud detection model which can display the detection effect of the model in real time and is also beneficial to analyze the specific application of the model. Figure 7 shows the number of intercepted electronic transactions of the detection model. While Figure 8 shows the performance indicators of the detection model running in the system, including hit rate, recall rate, accuracy rate and interference rate.

This chapter designs and implements a B/S-based group behavior fraud detection subsystem. The system mainly includes two functional modules: offline model training function and real-time detection of simulated electronic transactions. On the one hand, the offline model training module is used to access the API interface of the data storage platform of the hierarchical diagnosis and treatment cloud platform to obtain the historical transaction data of electronic transactions as the original training set. At the same time, the model parameters are set through visual interactive operations to realize electronic transactions based on deep forests.

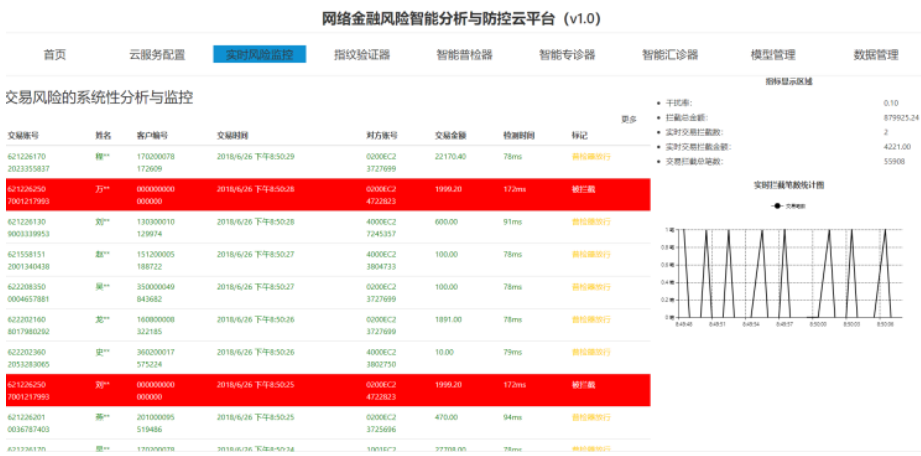


Figure 5. Transaction risk monitoring page

交易账号 Trading account	姓名 User name	客户编号 User ID	交易时间 Trading hours	对方账号 Reciprocal account number	交易金额 Transaction amount	检测时间 Detection time	标记 Test results
622202170 2026602508	万**	000000000 000000	2018/6/26 下午9:11:22	0213EC4 6895132	66.12	172ms	被拦截
622202170 2026602508	徐**	000000000 000000	2018/6/26 下午9:11:09	0213EC4 6895132	8.54	188ms	被拦截
622202170 2026602508	吴**	000000000 000000	2018/6/26 下午9:11:06	0213EC4 6895132	56.95	173ms	被拦截
622202170 2026602508	杨**	000000000 000000	2018/6/26 下午9:11:03	0213EC4 6895132	4.33	171ms	被拦截
622202170 2026602508	李**	000000000 000000	2018/6/26 下午9:10:54	0213EC4 6895132	66.12	172ms	被拦截
622202170 2026602508	毛**	000000000 000000	2018/6/26 下午9:10:51	0213EC4 6895132	8.54	172ms	被拦截
622202170 2026602508	万**	000000000 000000	2018/6/26 下午9:10:40	0213EC4 6895132	66.12	173ms	被拦截
622202170 2026602508	刘**	000000000 000000	2018/6/26 下午9:10:38	0213EC4 6895132	4.33	183ms	被拦截
622202170 2026602508	李**	000000000 000000	2018/6/26 下午9:10:29	0213EC4 6895132	4.33	174ms	被拦截

Figure 6. The set of transactions that the model identifies as fraudulent

Fraud model training and visualization. On the other hand, the real-time detection function of simulating electronic transactions is used to obtain the implementation transaction data stream of the risk control subsystem of the financial risk control platform and the detection model trained by the offline model training function is used to perform the real-time transaction data stream.

Real-time detection of fraudulent transactions and analysis of detection effects. By building a group behavior fraud detection subsystem, the development complexity of developers is reduced and the application value of the detection model proposed in this article is verified.

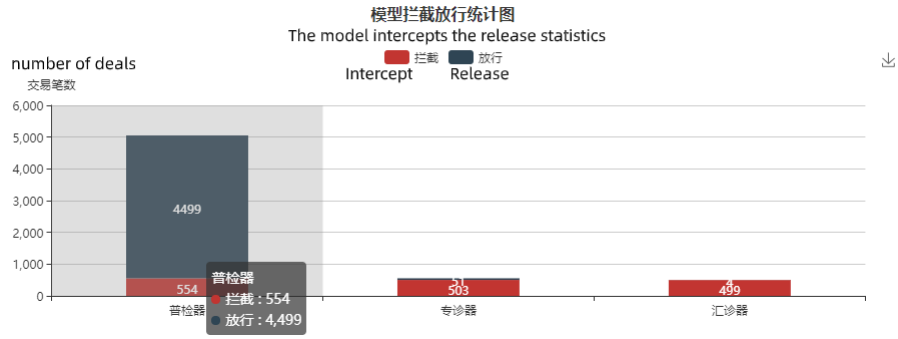


Figure 7. Each module intercepts the number of releases

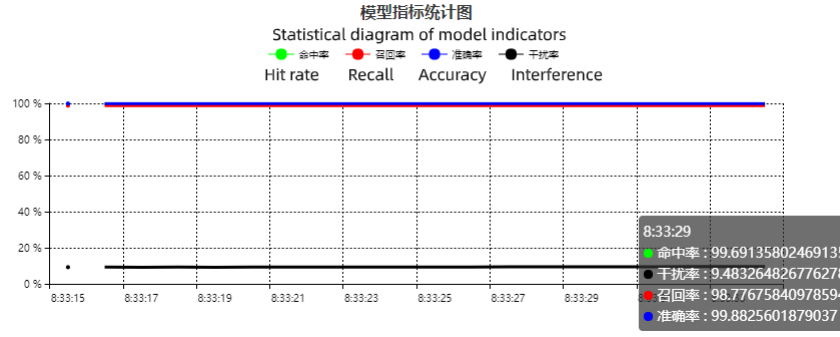


Figure 8. The model detects trade indicators in real time

5 CONCLUSION

This paper establishes an online transaction fraud detection model based on improved gcForest. BaggingBalance, a sample imbalance processing method based on Bagging, is proposed to rebalance the datasets and construct a global sample unbalance processing mechanism with XGBoost used in cascade forest. Based on this, autoencoder algorithm is introduced to the multi-scanning structure, further enhancing the representational learning ability of the model. The experimental results show a superior fraud detection performance of the proposed model on real bank online transaction data. Furthermore, this paper is another exploration about using the advantage of ML techniques and DL techniques. There are more possibilities for combining more ML models and DL models to detect online fraudulent transactions.

Acknowledgement

This work was supported by the Natural Science Foundation of Shanghai (No. 19ZR-1401900) and the Shanghai Science and Technology Innovation Action Plan Project (No. 19511101300).

REFERENCES

[1] Jingdong Financial Research Institute: Digital Finance Anti-Fraud White Paper. Available from: http://finance.qq.com/original/caijingzhiku/yzzk12.html, May 2018.

[2] Security Alliance of E-Commerce Ecosystem: 2017 E-Commerce Ecological Security White Paper. Available from: https://www.saee.org.cn/pc/newsContent/news20170726, July 2017.

[3] Sahin, Y.—Duman, E.: Detecting Credit Card Fraud by Decision Trees and Support Vector Machines. International MultiConference of Engineers and Computer Scientists, Vol. 1, 2011, pp. 442–447, doi: 10.1109/inista.2011.5946108.

[4] Alowais, M. I.—Soon, L. K.: Credit Card Fraud Detection: Personalized or Aggregated Model. Third FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing, June 2012, pp. 114–119, doi: 10.1109/music.2012.27.

[5] Xuan, S.—Liu, G.—Li, Z.—Zheng, L.—Wang, S.—Jiang, C.: Random Forest for Credit Card Fraud Detection. 15th International Conference on Networking, Sensing and Control, March 2018, pp. 1–6, doi: 10.1109/icnsc.2018.8361343.

[6] LeCun, Y.—Bengio, Y.—Hinton, G. E.: Deep Learning. Nature, Vol. 521, 2015, pp. 436–444, doi: 10.1038/nature14539.

[7] Krizhevsky, A.—Sutskever, I.—Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In: Pereira, F., Burges, C. J. C., Bottou, L., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 25 (NIPS 2012), 2012.

[8] Sutskever, I.—Vinyals, O.—Le, V. Q.: Sequence to Sequence Learning with Neural Networks. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 27 (NIPS 2014), Vol. 2, 2014, pp. 3104–3112.

[9] Corbo, J.—Giovine, C.—Wigley, C.: Applying Analytics in Financial Institutions Fight Against Fraud. McKinsey Analytics, April 2017. Available from: https://www.mckinsey.com/businessfunctions/mckinsey-analytics/our-insights/applying-analytics-infinancial-institutions-fight-against-fraud, retrieved February 2018.

[10] Li, X.—Yu, W.—Luwang, T.—Zheng, J.—Qiu, X.—Zhao, J. et al.: Transaction Fraud Detection Using GRU-Centered Sandwich-Structured Model. 2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design, November 2017, pp. 467–472, doi: 10.1109/cscwd.2018.8465147.

[11] Zhou, Z.—Feng, J.: Deep Forest: Towards an Alternative to Deep Neural Networks. Proceedings of the Twenty-Sixth International Joint Conference on Arti-

ficial Intelligence (IJCAI 2017), February 2017, pp. 3553–3559, doi: 10.24963/ijcai.2017/497.

[12] Breiman, L.: Random Forests. Machine Learning, Vol. 45, 2001, No. 1, pp. 5–32, doi: 10.1023/A:1010933404324.

[13] Liu, F.—Ting, K.—Yu, Y.—Zhou, Z.: Spectrum of Variable-Random Trees. Journal of Artificial Intelligence Research, Vol. 32, 2008, pp. 355–384, doi: 10.1613/jair.2470.

[14] Dong, M.—Yao, L.—Wang, X.—Benatallah, B.—Huang, C.—Ning, X.: Opinion Fraud Detection via Neural Autoencoder Decision Forest. Pattern Recognition Letters, Vol. 132, 2020, pp. 21–29, doi: 10.1016/j.patrec.2018.07.013.

[15] Chen, T.—Guestrin, C.: XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[16] Nielsen, D.: Tree Boosting with XGBoost – Why Does XGBoost Win "Every" Machine Learning Competition? Master's Thesis, Norvegian University of Science and Technology (NTNU), Trondheim, 2016.

[17] Olszewski, D.: Fraud Detection Using Self-Organizing Map Visualizing the User Profiles. Knowledge-Based Systems, Vol. 70, 2014, pp. 324–334, doi: 10.1016/j.knosys.2014.07.008.

[18] Dal Pozzolo, A.: Adaptive Machine Learning for Credit Card Fraud Detection. Ph.D. Thesis, Université Libre de Bruxelles, December 2015.

[19] Fu, K.—Cheng, D.—Tu, Y.—Zhang, L.: Credit Card Fraud Detection Using Convolutional Neural Networks. In: Hirose, A., Ozawa, S., Doya, K., Ikeda, K., Lee, M., Liu, D. (Eds.): Neural Information Processing (ICONIP 2016). Springer, Cham, Lecture Notes in Computer Science, Vol. 9949, 2016, pp. 483–490, doi: 10.1007/978-3-319-46675-0_53.

[20] Wang, S.—Liu, C.—Gao, X.—Qu, H.—Xu, W.: Session-Based Fraud Detection in Online E-Commerce Transactions Using Recurrent Neural Networks. In: Altun, Y. et al. (Eds.): Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2017). Springer, Cham, Lecture Notes in Computer Science, Vol. 10536, 2017, pp. 241–252, doi: 10.1007/978-3-319-71273-4_20.

[21] Zhang, Z.—Zhou, X.—Zhang, X.—Wang, L.—Wang, P.: A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection. Security and Communication Networks, Vol. 2018, 2018, Art. No. 5680264, 9 pp., doi: 10.1155/2018/5680264.

[22] Roy, A.—Sun, J.—Mahoney, R.—Alonzi, L.—Adams, S.—Beling, P.: Deep Learning Detecting Fraud in Credit Card Transactions. 2018 Systems and Information Engineering Design Symposium (SIEDS), 2018, pp. 129–134, doi: 10.1109/SIEDS.2018.8374722.

[23] Kazemi, Z.—Zarrabi, H.: Using Deep Networks for Fraud Detection in the Credit Card Transactions. 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), 2017, pp. 630–633, doi: 10.1109/kbei.2017.8324876.

[24] Breiman, L.: Bagging Predictors. Machine Learning, Vol. 24, 1996, No. 2, pp. 123–140, doi: 10.1007/bf00058655.

Mian Huang received the M.Sc. degree from Lanzhou University of Technology, Lanzhou, China in 2008, and now he is pursuing the Eng.D. degree from Tongji University, Shanghai, China. His current research interests include network security and identity authentication.

Lizhi Wang is an M.Sc. candidate at the Donghua University. His research area includes machine learning, deep learning, and cloud computing.

Zhaohui Zhang obtained his Bachelor's degree in computer science from Anhui Normal University, Wuhu, China in 1994. He obtained his Ph.D. in computer science from Tongji University, Shanghai, China in 2007. From 1994 to 2015, he worked in Anhui Normal University as Professor. Since 2015 he has been working as Professor in the School of Computer Science and Technology, Donghua University, Shanghai, China. His research interests include big data intelligent processing and behavior analysis.

Computing and Informatics, Vol. 39, 2020, 1099–1116, doi: 10.31577/cai_2020_5_1099

FAULT TOLERANCE IN REVERSIBLE LOGIC CIRCUITS AND QUANTUM COST OPTIMIZATION

Kamaraj Arunachalam

Department of Electronics and Communication Engineering
Mepco Schlenk Engineering College
Sivakasi, Tamilnadu, India
e-mail: kamarajvlsi@gmail.com

Marichamy Perumalsamy

Department of Electronics and Communication Engineering
PSR Engineering College
Sivakasi, Tamilnadu, India
e-mail: pmarichamy@psr.edu.in

Kaviyashri K. Ponnusamy

Department of Electronics and Communication Engineering
Mepco Schlenk Engineering College
Sivakasi, Tamilnadu, India
e-mail: kaviyashrikp@gmail.com

Abstract. Energy dissipation is a prominent factor for the very large scale integrated circuit (VLSI). The reversible logic-based circuit was capable to compute the logic without energy dissipation. Accordingly, reversible circuits are an emerging domain of research based on the low value of energy dissipation. At nano-level design, the critical factor in the logic computing paradigm is the fault. The proposed methodology of fault coverage is powerful for testability. In this article, we target three factors such as fault tolerance, fault coverage and fault detection in the reversible KMD Gates. Our analysis provides good evidence that the minimum test vector covers the 100 % fault coverage and 50 % fault tolerance in KMD Gate.

Further, we show a comparison between the quantum equivalent and controlled V and V^+ gate in all the types of KMD Gates. The proposed methodology mentions that after controlled V and V^+ gate based ALU, divider and Vedic multiplier have a significant reduction in quantum cost. The comparative results of designs such as Vedic multiplier, division unit and ALU are obtained and they are analyzed showing significant improvement in quantum cost.

Keywords: KMD Gate, controlled V and V^+ gate, ALU, divider and Vedic multiplier

1 INTRODUCTION

Launder proved that any irreversible or conventional computation will dissipate $KTln2$ Joules (k is Boltzmann's constant and T is the temperature) of heat since it loses bit information from input to output transition [1]. Bennett introduced a logically reversible computing machine [2].

In reversible logic, the energy dissipation during computation is null, since it has a bijective mapping between input and output. The reversible logic circuits are constructed using a distinguished well-defined library of gates. Good reversible logic circuits must be well optimized in terms of quantum cost (QC), constant input (CI), garbage output (GO) and logical calculations (LC) [3]. The reversible logic circuits are established in quantum cellular automata and optical computation.

The testing of circuits will guarantee the perfect operation. Generally, two major classification of testing: which are online and offline. Testable circuits have their challenges such as test data minimization, low-level signals, variety of fault models [4].

Initially, reversible logic gates/circuits testing in online/offline over the benchmark circuits have been done in [5]. It has been extended towards defining the performance measure parameters like Missing Gate, Cell fault, Stuck-at Fault. Also, two testable reversible gates R1 and R2 are presented for online testing [6].

Thereafter, many researchers had contributed to the reversible fault tolerant architectures such as adders, ALU and floating-point units. Variety of Adders (CSA, RCA, BCD and CLA) are constructed using ZCG, LCG, MIG, Fredkin and F2G parity preserving gates. It is observed that the parity preserving gates provides improved performance in the various adder structures [7]. In RUG gate fault pattern based Fault tolerance is analyzed and it has 52.2 of average fault tolerance on faults. An ALU has been constructed using this reversible gate [8].

Moreover, the synthesis of reversible circuits is a significant part of optimizing performance measures. The quantum reversible circuits are obtained for the basic gates. The quantum circuits can be derived from mapping them to the NCV library [9]. Also, fault models and various approaches towards test pattern generation are discussed here in [9].

In this paper, the fault coverage, fault tolerance and fault detection test vector for the KMD Gates are discussed. Then arithmetic and logic circuit, floating-point division and Vedic multipliers are constructed using KMD Gates. These reversible circuits are optimized in quantum cost using the behavioral model of integrated qubit optimization [10].

The rest of the paper is organized as follows: Section 2 deals with fundamental fault models in reversible logic with an example. Section 3 describes the methods used to reduce the quantum cost in detail; then in Section 4 fault tolerance in KMD Gates is checked. Thereafter, in Section 5, quantum cost optimization in arithmetic circuits such as ALU, division and Vedic multiplier is elaborated. Finally, the conclusion is presented with future scope.

2 PRELIMINARIES OF FAULT MODELS IN REVERSIBLE LOGIC

Faults are any types of imperfection in a system that affects the functional behavior of a system either permanently or temporarily. The fault is caused either by manual or environmental factors. A fault model describes the type of fault that occurred in the system and it identifies the target of testing. There are many types of fault methods [4]:

- Stuck-at fault,
- Bridging fault,
- Missing gate fault,
- Cell fault,
- Cross-point fault.

2.1 Stuck-At Fault

In the stuck-at fault model, the fault occurred in a circuit when any wire fixed on a value '0' or '1', called as stuck-at 0 or stuck-at 1 fault, respectively. Total number of stuck-at faults can be obtained, as shown in Equation (1) [4]

$$2\left(N+\sum_{i=1}^{m} g_i\right) \tag{1}$$

where g_i represents the size of the N^{th} gate of the circuit, N represents the total number of wires and m represents the number of gates in the circuit.

2.2 Minimum Test Vectors

The minimum test vectors are the test vectors covering maximum faults occurring in the reversible circuit.

2.3 Fault Coverage

It is defined as the ratio of the actual number of detected faults to the total number of faults present in the circuits [11]. Based on fault coverage, the efficiency of the testing techniques can be explained as in [11, 12].

fault coverage = (number of detected fault)/(number of detectable fault).

2.4 Fault Tolerance

Fault tolerance is the property of the system that permits a system to work continuously even in the failure of some of its components. The reversible gates with parity preservation are also known as fault tolerant gates [13].

2.5 Controlled V and Controlled V^+ Gates

The V gate is the square root of NOT gate and the V^+ gate is the Hermitian conjugate of the V gate [3, 10]. The V and V^+ gates have the following properties [5, 6]:

$$\begin{aligned} V \times V &= \text{NOT}, \\ V \times V^+ &= V^+ \times V = I, \\ V^+ \times V^+ &= \text{NOT}. \end{aligned} \tag{2}$$

In order to construct a truth table for V and V^+ gates, the properties of these gates are used, as proposed in Equation (2) [3]. This equation shows that when two V gates or two V^+ gates are in series it is equal to a NOT gate. Likewise, when one V and V+ gates are in series, its logical equivalent is identity.

3 QUANTUM COST OPTIMIZATION IN KMD GATES

The Quantum Cost of the reversible logic gate/circuit can be reduced by removing the redundant gates in the quantum equivalent circuits and/or combining the gates in the controlled V and V^+ Structure. At first, the quantum equivalent gate/circuit is obtained from Toffoli-Fredkin Code using the desired expression of the gate/circuit. Then it is decomposed into controlled V and V^+ gate. It is a composition of 2×2 and 1×1 gate of the reversible gate/circuit. The Quantum cost is the sum of the number of 2×2 and 1×1 gates present in the decomposed structure. It is then applied with integrated qubit optimization to remove the redundant gates in the decomposed V and V^+. The process is repeated until further optimization is not possible. The complete process flow graph is shown in Figure 1.

The quantum cost of some of the configurations of the V and V^+ structure is one as shown in Table 1. So, in the controlled V and V^+ structure, wherever these combinations are present it would be considered as one quantum cost. For example,

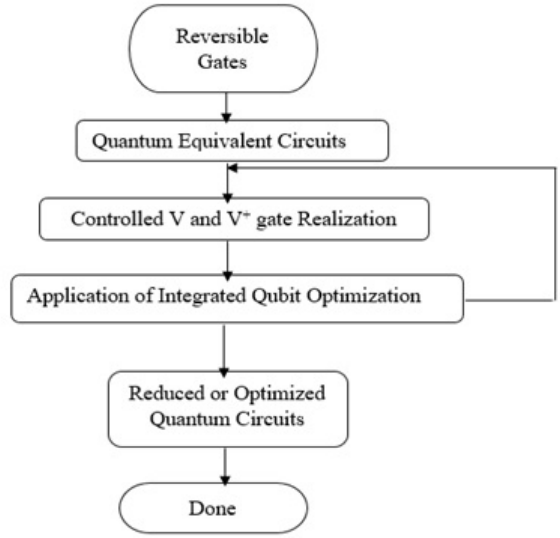


Figure 1. Process flow chart of quantum cost reduction of reversible circuit

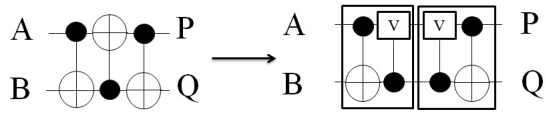


Figure 2. a) C-NOT gates, b) C-NOT gate using controlled V gate

The two controlled-V gates are used instead of single NOT gate. In Figure 2 a) the quantum cost is 3, by rearranging or restructuring using controlled-V gates the quantum cost is reduced to 2. By using this method the quantum cost of any circuit can be reduced. Also, the controlled V and V^+ structures have a reduction in quantum cost in reversible gates and circuits according to the integrated qubit rules [10] are shown in Table 1.

Also, when either two consecutive NOT gate or consecutive controlled V and V^+ gate is present the quantum cost will be zero [3], as shown in Figure 3.

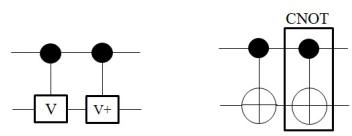


Figure 3. Equivalent circuit having quantum cost as zero

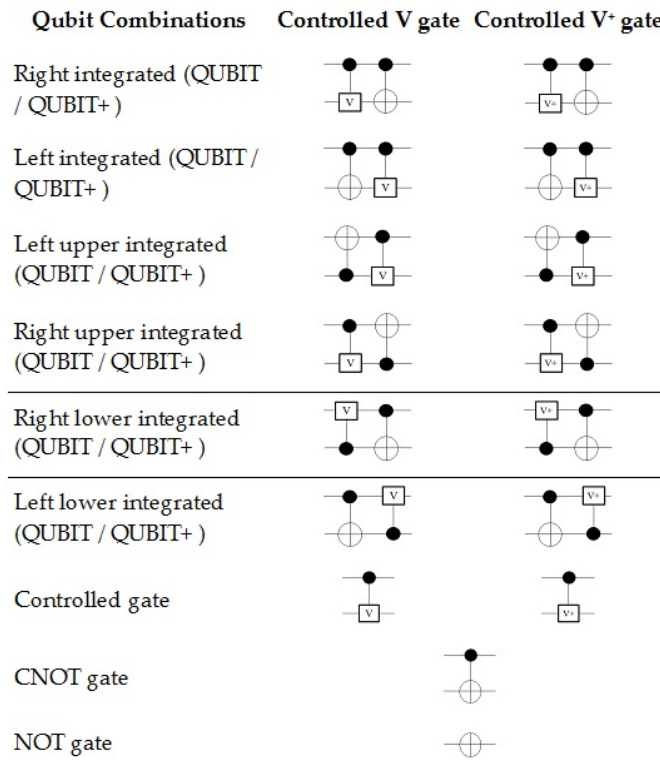


Table 1. Quantum structure of reversible gates with quantum cost = 1 [10]

Using the above two principles the quantum cost of the KMD Gates [14, 15] are optimized as follows.

3.1 KMD Gate1

It is a 3 × 3 gate. In quantum circuit A, B, C represents the input and P, Q, R represents the output. Here in Figure 4 c), there is 9 number of 2 × 2 gates, but according to the constraints [10, 17], the two qubits can be combined. Therefore the quantum cost is 8.

3.2 KMD Gate2

It is a 3 × 3 gate. In quantum circuit A, B, C represents the input and P, Q, R represents the output. Here in Figure 5 c), there is 10 number of 2 × 2 gates, but

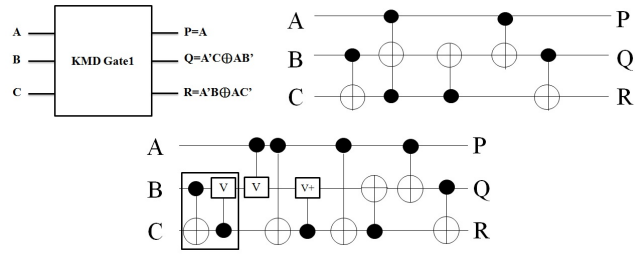


Figure 4. a) Block diagram, b) quantum equivalent, c) V and V^+ gate realization

according to the constraints [10, 17], the two qubits can be combined. Therefore the quantum cost is 9.

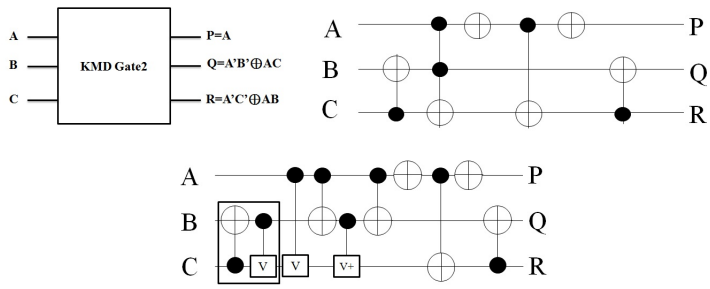


Figure 5. a) Block diagram, b) quantum equivalent, c) V and V^+ gate realization

3.3 KMD Gate3

It is a 4×4 gate. In the quantum circuit, A, B, C, D represents the input and P, Q, R, S represents the output. Here in Figure 6 c), there is 9 number of 2×2 gates, but according to the constraints [10, 17], the two qubits can be combined. Therefore the quantum cost is 8.

3.4 KMD Gate4

It is a 5×5 gate. In the quantum circuit, A, B, C, D, E represents the input and P, Q, R, S, T represents the output. Here in Figure 7 b) there is 24 number of 2×2 gates, but according to rules the two constructive CNOT gate has quantum cost as zero (i.e., canceled). Therefore the quantum cost is 20.

The controlled V and V^+ have a reduction in the quantum cost of reversible gates according to the integrated qubit rules stated in Table 1 using the concepts of [10, 17]. The minimized quantum cost is shown in Table 2. In KMD Gates the quantum cost is reduced in the range of 10 % to 16.67 %.

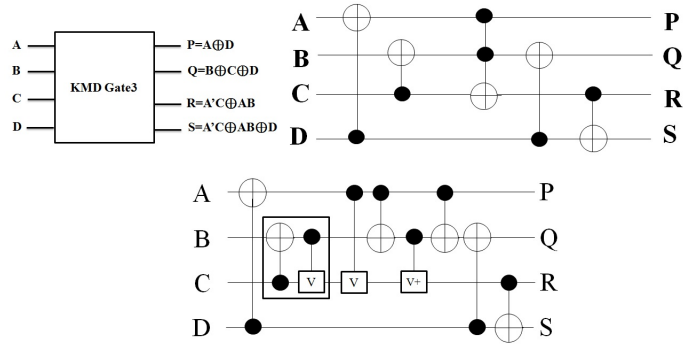


Figure 6. a) Block diagram, b) quantum equivalent, c) V and V^+ gate realization

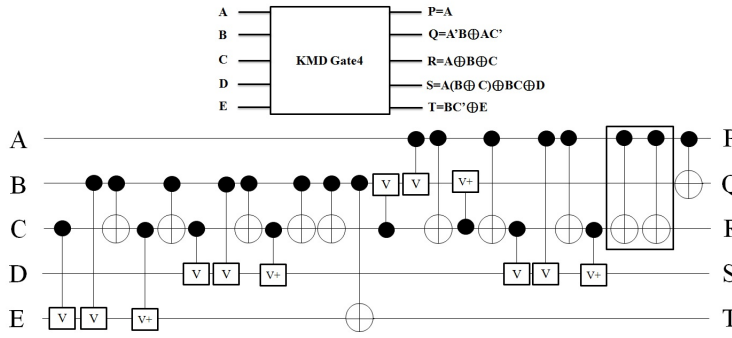


Figure 7. a) Block diagram, b) V and V^+ gate realization

4 FAULT TOLERANCE IN KMD GATES AND CIRCUITS

A reversible gate is said to be parity preserving when the result of XOR operation on input vectors $I_v = I_1, I_2, \ldots, I_n$ is the same as that of the XOR operation on output vectors $O_v = O_1, O_2, \ldots, O_n$, as represented in Equation (3) [7].

Gates	Existing Method [14]	V and V+ gate Realization	% of minimization
KMD Gate1	9	8	11.11%
KMD Gate2	10	9	10%
KMD Gate3	9	8	11.11%
KMD Gate4	24	20	16.67%
MNFT	7	6	14.28%
ZPLG	12	10	16.67%

Table 2. Comparison between quantum equivalent circuit and controlled-V and V^+

$$I_1 \oplus I_2 \oplus \cdots \oplus I_{n-1} \oplus I_n = O_1 \oplus O_2 \oplus \cdots \oplus O_{n-1} \oplus O_n. \quad (3)$$

The test vectors of the circuit or gate are identified using fault table analysis. Using this table, an input capable of covering the maximum number of fault is chosen as the Test vector. A set of test vectors which covers all the fault is known as Test set [13]. This test set is used to check the functionality of the reversible gate whether it is good or faulty. Fault coverage and fault tolerance are estimated for the reversible gate/circuit as shown in Figure 8.

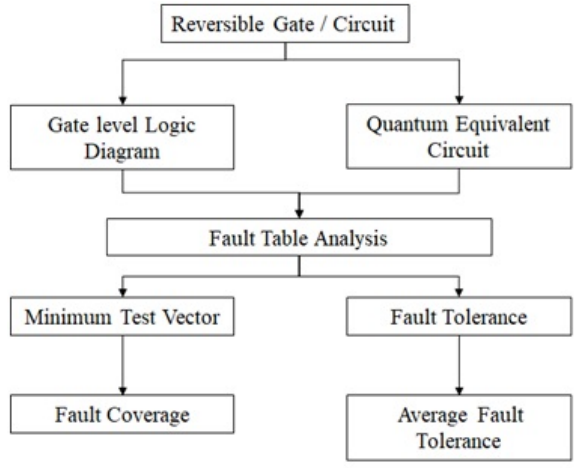


Figure 8. Fault tolerance analysis flow chart

The fault table of the KMD Gate1 is shown in Table 3. The '*' indicates the possible faults that can be detected using the corresponding input test vector. The minimum test vectors to detect all the possible faults are identified from the fault table.

Inputs			Faults																								
A	B	C	a/0	a/1	b/0	b/1	c/0	c/1	d/0	d/1	e/0	e/1	f/0	f/1	g/0	g/1	h/0	h/1	i/0	i/1	j/0	j/1	k/0	k/1	l/0	l/1
0	0	0		*	*		*			*	*			*		*	*		*			*		*		*
0	0	1		*	*		*			*		*		*	*		*		*			*	*			*
0	1	0		*	*			*		*	*		*			*	*		*		*			*	*	
0	1	1		*		*	*		*		*			*	*		*		*		*		*		*	
1	0	0	*			*	*		*			*		*	*		*		*		*		*		*	
1	0	1	*		*		*		*			*		*	*		*		*			*	*			*
1	1	0	*			*		*	*			*	*		*			*	*		*			*	*	
1	1	1	*			*		*	*			*	*		*			*		*	*			*		*
			4	4	4	4	5	3	5	3	3	5	3	5	6	2	6	2	7	1	5	3	4	4	4	4

Table 3. Fault table of KMD Gate1

4.1 Minimum Test Vectors

$$000 - a/1b/0c/0d/1e/0f/1g/1h/0i/0j/1k/1l/1,$$
$$011 - a/1b/1c/0d/0e/0f/1g/0h/0i/0j/0k/0l/0,$$
$$111 - a/0b/1c/1d/0e/1f/0g/0h/1i/1j/0k/1l/1.$$

Here, the three minimum test vectors cover the entire fault in the KMD Gate1. Thus, to detect the fault in the gate;

$$\text{Test Vector} = 000, 011, 111.$$

4.2 Fault Coverage

After removing redundancy from the above minimum test vectors, the possible number of faults can be detected:

$$000 - A/1B/0C/0P/1Q/1R/1 = 6/12,$$
$$011 - Q/0R/0 = 2/12,$$
$$111 - A/0B/1C/1P/0 = 4/12.$$

Stuck-at- faults	Testing position	Test input			Target output			Faulty output		
S-A-0	A	1	1	1	1	0	0	0	1	1
	B	1	1	1	1	0	0	1	1	0
	C	1	1	1	1	0	0	1	0	1
	P	1	1	1	1	0	0	0	0	0
	Q	0	1	1	0	1	1	0	0	1
	R	0	1	1	0	1	1	0	1	0
S-A-1	A	0	0	0	0	0	0	1	1	1
	B	0	0	0	0	0	0	0	0	1
	C	0	0	0	0	0	0	0	1	0
	P	0	0	0	0	0	0	1	0	0
	Q	0	0	0	0	0	0	0	1	0
	R	0	0	0	0	0	0	0	1	1

Table 4. Fault coverage table of KMD Gate1

Hence, all the faults in the KMD Gate1 can be found using the three Test vectors (000, 011 and 111) and their cumulative fault coverage is 100 % as from Table 4. The average fault tolerance of KMD Gate1 is 50 % as shown in Table 5.

ABC	a/0	a/1	b/0	b/1	c/0	c/1	d/0	d/1	e/0	e/1	f/0	f/1	g/0	g/1	h/0	h/1	i/0	i/1	j/0	j/1	k/0	k/1	l/0	l/1	m/0	m/1	n/0	n/1	Fault coverage
000	a0	a7	a0	a1	a0	a2	a0	a3	a0	a2	a0	a7	a0	a3	a0	a2	a0	a3	a0	a1	a0	a4	a0	a3	a0	a2	a0	a1	14/28
001	a2	a6	a2	a3	a0	a2	a2	a1	a0	a2	a2	a5	a2	a1	a0	a2	a1	a2	a3	a2	a2	a6	a1	a2	a0	a2	a2	a3	14/28
010	a1	a5	a0	a1	a1	a3	a2	a1	a3	a1	a1	a6	a2	a1	a3	a1	a1	a2	a0	a1	a1	a5	a1	a2	a1	a3	a0	a1	14/28
011	a3	a4	a2	a3	a1	a6	a0	a3	a3	a1	a3	a4	a0	a3	a3	a1	a0	a3	a3	a2	a3	a7	a0	a3	a1	a3	a2	a3	14/28
100	a0	a7	a7	a5	a7	a6	a7	a4	a7	a6	a0	a7	a7	a4	a7	a5	a7	a4	a7	a6	a3	a7	a4	a7	a5	a7	a6	a7	14/28
101	a2	a6	a6	a4	a7	a6	a6	a5	a7	a6	a1	a6	a5	a6	a4	a6	a6	a5	a7	a6	a2	a6	a5	a6	a4	a6	a6	a7	14/28
110	a1	a5	a7	a5	a5	a4	a6	a5	a4	a5	a2	a5	a5	a6	a7	a5	a6	a5	a4	a5	a1	a5	a5	a7	a5	a7	a4	a5	14/28
111	a3	a4	a6	a4	a5	a4	a7	a4	a4	a5	a3	a4	a7	a4	a4	a6	a7	a4	a4	a5	a0	a4	a4	a6	a4	a6	a4	a5	14/28

Table 5. Fault tolerant table of KMD Gate1

Similarly, it is possible to found out Test vector, fault coverage and fault tolerance for the KMD Gate2, KMD Gate3 and KMD Gate4 as in Table 6. From the table, it is observed that the fault coverage is 100 % for all the KMD Gates and average fault tolerance is approximately 50 %.

The complete processing steps for all the four KMD Gates are made available at online repository (`https://github.com/kamarajvlsi/Reversible_Logic`).

Reversible Gate	Test vectors	Fault Coverage (After redundancy removal)	Average Fault Tolerance
KMD Gate1	000, 011, 111	000 – A/1 B/0 C/0 P/1 Q/1 R/1 = 6/12 011 – Q/0 R/0 = 2/12 111 – A/0 B/1 C/1 P/0 = 4/12	50%
KMD Gate2	000, 100, 011	000 – A/1 B/1 P/1 Q/0 R/0 = 5/12 100 – A/0 B/0 C/0 P/0 Q/1 R/1 = 6/12 011 – C/1 = 1/12	47.32%
KMD Gate3	0000, 0111, 1011	0000 – A/1 B/1 C/0 D/0 P/1 Q/1 R/1 S/1 = 8/16 0111 – A/0 P/0 Q/0 S/0 = 4/16 1011 – B/0 C/1 D/1 R/0 = 4/16	45.42%
KMD Gate4	00000, 00100, 01100, 10111	00000 – A/1 D/1 E/1 P/1 Q/1 R/1 S/1 T/1 = 8/20 00100 – B/1 C/0 D/0 E/0 R/0 = 5/20 01100 – C/1 Q/0 S/0 = 3/20 10111 – A/0 P/0 T/0 = 5/20	40.25%

Table 6. Fault analysis of KMD Gates

5 QUANTUM COST OPTIMIZATION IN ARITHMETIC CIRCUITS

5.1 Arithmetic and Logic Unit

The arithmetic and logic unit constructed in the [14] using the KMD Gates performs 18 distinct operations. The architecture consists of logic gates, adders and multiplexers, as shown in Figure 9. It has a minimum number of control signals and the integrated architecture performs both arithmetic and logical operations in the same structure. In that, two approaches were followed; one is constructing ALU

using KMD Gates alone and another is using KMD, Toffoli and Fredkin Gates. In both approaches, parity preservation is maintained.

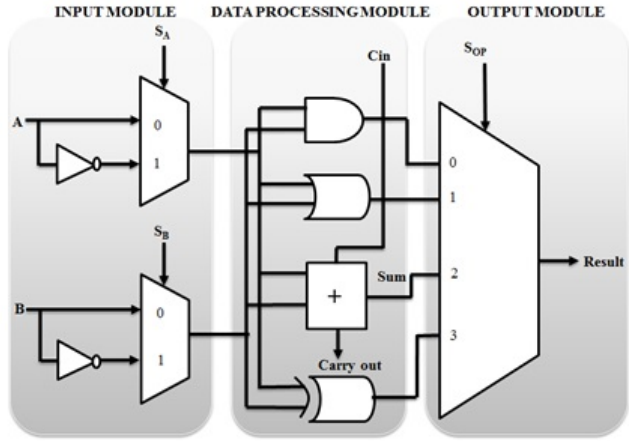


Figure 9. ALU architecture [14]

The overall quantum cost of the integrated ALU is reduced, after applying Integrated qubit optimization as shown in Table 7.

Architecture	ALU (Before V and V+) [14]		ALU (After V and V+ realization)		% of improvement	
Methods	Approach 1	Approach 2	Approach 1	Approach 2	Approach 1	Approach 2
Quantum Cost	116 n	100 n	101 n	92 n	13%	8%
Constant Input	7 n	8n	7n	8n	-	-
Garbage Output	21n	22n	21n	22n	-	-
Number of Gates	64	46	64	46	-	-
Total Cost (QC+CI+GO+NoG)	205	176	193	168	5.85%	4.54%
Logical Calculation	12α + 14β +6??	12α + 14β +6??	12α + 14β +6??	12α + 14β +6??	-	-

Table 7. Quantum cost of ALU before and after controlled-V and V^+ gate realization

5.2 Floating Point Division Unit

The floating-point (FP) operation is a time consuming one in the processor. In [15] a floating-point division unit is proposed with IEEE 754 single-precision format using a non-restoring algorithm. The n-bit FP division consists of the multiplexer, parallel adder and registers. The multi-function register performs, serial-in, parallel in and hold operations as shown in Figure 10.

The percentage of improvement in quantum cost after V and V^+ structure optimization is shown in Tables 8 and 9.

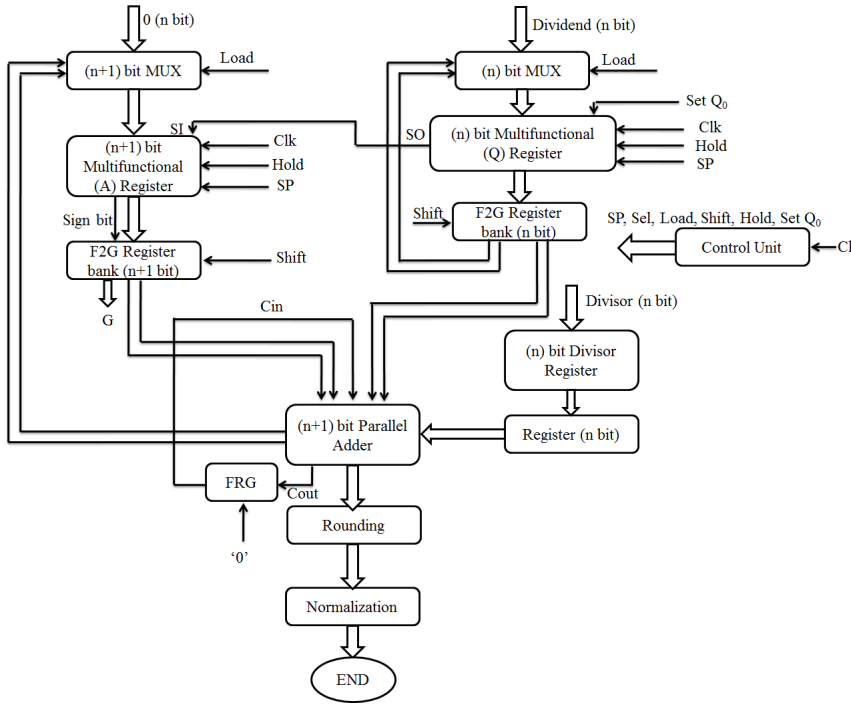


Figure 10. Fault-tolerant floating-point division unit [15]

Modules	No of bits	Gates used	Quantum Cost: Before applying Control V and V+ [15]	Quantum Cost: After applying Control V and V+	% of improvement in Quantum Cost
MUX	n	KMD Gate3	9n	8n	11.12%
	n+1		9n+9	8n+8	11.12%
Multifunctional register	3n		28n	25n	10.72%
	3n+1		28n+28	25n+25	10.72%
Divisor Register	n	F2G	2n	2n	-
Parallel Adder	n+1	KMD Gate4	22n+22	20n+20	9.1%
Register	n	F2G	2n	2n	-
	n+1		2n+2	2n+2	-
Rounding	n+1	KMD Gate3 + F2G	9n+9+2n+2	8n+8+2n+2	11.12%
Normalization	n+1	KMD Gate3	9n+9	8n+8	11.12%
Total Cost of Division			122n+81	110n+73	9.86%

Table 8. Quantum cost of the division unit before and after controlled-V and V^+ gate realization (for n-bit)

No. of bits	Before applying Control V and V+ [15]	After applying Control V and V+	% of reduction in Quantum Cost
1	203	183	9.85%
2	325	293	9.84%
4	569	513	9.84%
8	1,057	953	9.84%
16	2,033	1,833	9.84%
32	3,985	3,593	9.84%
64	7,889	7,113	9.83%
128	15,697	14,153	9.83%
256	31,313	28,233	9.83%

Table 9. Quantum cost of the division unit (1–256 bits)

5.3 Vedic Multiplier (VM)

The 2×2 Vedic multiplier can be constructed using 4 AND gates and 2 half adder, as shown in Figure 11. A 2-bit multiplication of two numbers A.B could be carried out in the following manner, as shown in Equation (4). The logical expression of the 2×2 Vedic multiplier final product is

$$\begin{aligned} P0 &= A0.B0, \\ P1 &= (A1.B0) \oplus (A0.B1), \\ P2 &= (A0.A1.B0.B1) \oplus (A1.B1), \\ P3 &= A0.A1.B0.B1. \end{aligned} \tag{4}$$

The AND gates are constructed using the KMD Gate2 and half adder structure is constructed using KMD Gate3. By fixing $C = 0$ and $D = B$ in the KMD Gate3 the half adder circuit is obtained and the results are taken from P and R in the output side [16].

The quantum cost of the 4×4 Vedic multiplier as shown in Figure 11 is 84 and the constant input is 48. Here the output other than P0, P1, ..., P7 is considered as the garbage output [16]. In order to reduce the quantum cost of the circuit, the controlled V and V+ gate realization is performed as shown in Table 10. After applying integrated qubit principles, the quantum cost of the 4-bit VM is reduced by 10 % and it has an impact on the total cost reduction as 7.4 %.

The complete process and methodology, functional descriptions, simulations in QCA environment and analysis procedure are made available in online repository (`https://github.com/kamarajvlsi/Reversible_Logic`).

6 CONCLUSION

Reversible logic based computing systems consume ideally zero power dissipation. At nano-metric circuit design, the fault detection and fault-tolerant are significant

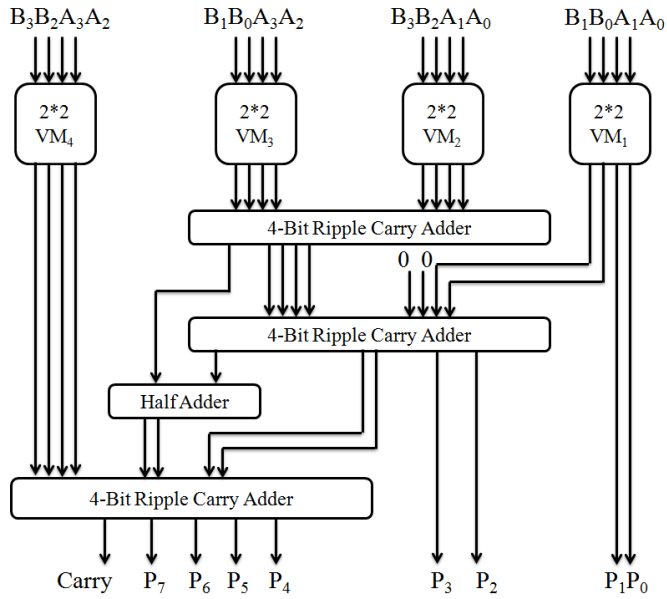


Figure 11. The hardware structure of 4-bit Vedic multiplier [16]

parameters. In the proposed methodology, it has been focused on fault tolerance, fault coverage and fault detection in the reversible KMD Gates. Our analysis provides evidence that minimum test vectors cover the 100 % fault coverage and 50 % fault tolerance in KMD Gates. Further, comparisons between the quantum equivalent and controlled V-V^+ gate for all the types of KMD Gates are shown. The

Module	Garbage Output	Constant Input	No. of Primitive Gates	No. of Gates	Quantum Cost (applying Control V and V⁺ structure)		% of Improvement in Quantum Cost
					Before [16]	After	
AND Gate	2	1	6	1	9	8	11.12%
Half Adder	2	1	5	1	9	8	11.12%
Full Adder	3	2	8	1	22	20	9.1%
2-bit Vedic Multiplier	12	6	34	6	54	48	11.12%
4-bit Ripple Carry Adder	12	8	32	4	88	80	9.1%
4-bit Vedic Multiplier	84	48	232	36	480	432	10%
Total Cost (QC+GO + CI + NoG)					648	600	7.4%

Table 10. Quantum cost of the Vedic multiplier before and after controlled V and V^+ gate realization

controlled V-V+ has been applied to ALU, division and multiplier. It has been observed that the quantum cost has reduced 8–13 %, 9.8 % and 10 % in ALU, floating point division and Vedic multiplier, respectively. It has a good impact on the total cost reduction of 4.5–5.8 % and 7.4 % in ALU and Vedic multiplier. So, we conclude that the quantum cost of the reversible logic circuit can be reduced by applying controlled V–V^+ on them. Further, this work can be incorporated in a processor design in a nano-metric level.

REFERENCES

[1] Landauer, R.: Irreversibility and Heat Generation in the Computing Process. IBM Journal of Research and Development, Vol. 5, 1961, No. 3, pp. 183–191, doi: 10.1147/rd.53.0183.

[2] Bennett, C. H.: Logical Reversibility of Computation. IBM Journal of Research and Development, Vol. 17, 1973, No. 6, pp. 525–532, doi: 10.1147/rd.176.0525.

[3] Misra, N. K.—Sen, B.—Wairya, S.—Bhoi, B.: Testable Novel Parity-Preserving Reversible Gate and Low-Cost Quantum Decoder Design in 1D Molecular-QCA. Journal of Circuits, Systems, and Computers, Vol. 26, 2017, No. 9, Art. No. 1750145, pp. 1–26, doi: 10.1142/s0218126617501456.

[4] Gaur, H. M.—Singh, A. K.—Ghanekar, U.: Offline Testing of Reversible Logic Circuits: An Analysis. Integration. VLSI Journal, Vol. 62, 2018, pp. 50–67, doi: 10.1016/j.vlsi.2018.01.004.

[5] Al Mamun, M. S.—Mondal, P. K.—Prodhan, U. K.: A Novel Approach for Designing Online Testable Reversible Circuits. International Journal of Engineering Research and Development, Vol. 5, 2012, No. 2, pp. 39–44.

[6] Gaur, H. M.—Singh, A. K.—Ghanekar, U.: A Review on Online Testability for Reversible Logic. Procedia Computer Science, Vol. 70, 2015, pp. 384–391, doi: 10.1016/j.procs.2015.10.041.

[7] Valinataj, M.—Mirshekar, M.—Jazayeri, H.: Novel Low-Cost and Fault-Tolerant Reversible Logic Adders. Computers and Electrical Engineering, Vol. 53, 2016, pp. 56–72, doi: 10.1016/j.compeleceng.2016.06.008.

[8] Sasamal, T. N.—Mohan, A.—Singh, A. K.: Efficient Design of Reversible Logic ALU Using Coplanar Quantum-Dot Cellular Automata. Journal of Circuits, Systems, and Computers, Vol. 27, 2018, No. 2, Art. No. 1850021, pp. 1–19, doi: 10.1142/S0218126618500214.

[9] Wille, R.—Chattopadhyay, A.—Drechsler, R.: From Reversible Logic to Quantum Circuits: Logic Design for an Emerging Technology. 2016 IEEE International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), 2016, pp. 268–274, doi: 10.1109/samos.2016.7818357.

[10] Lewandowski, M.—Ranganathan, N.—Morrison, M.: Behavioral Model of Integrated Qubit Gates for Quantum Reversible Logic Design. IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2013, pp. 194–199, doi: 10.1109/isvlsi.2013.6654658.

[11] Misra, N. K.—Wairya, S.—Sen, B.: Design of Conservative, Reversible Sequential Logic for Cost Efficient Emerging Nano Circuits with Enhanced Testability. Ain Shams Engineering Journal, Vol. 9, 2018, No. 4, pp. 2027–2037, doi: 10.1016/j.asej.2017.02.005.

[12] Thilak, K. R.—Gayathri, S.: Fault Coverage Analysis Using Fault Model and Functional Testing for DPM Reduction. IEEE International Conference on Emerging Research in Electronics, Computer Science and Technology (ICERECT), 2015, pp. 76–81, doi: 10.1109/erect.2015.7498991.

[13] Babu, H. M. H.—Mia, M. S.: Design of a Compact Reversible Fault Tolerant Division Circuit. Microelectronics Journal, Vol. 51, 2016, pp. 15–29, doi: 10.1016/j.mejo.2016.01.003.

[14] Kamaraj, A.—Marichamy, P.: Design of Integrated Reversible Fault-Tolerant Arithmetic and Logic Unit. Microprocessors and Microsystems, Vol. 69, 2019, pp. 16–23, doi: 10.1016/j.micpro.2019.05.009.

[15] Kamaraj, A.—Marichamy, P.: Design of Fault-Tolerant Reversible Floating Point Division. Journal of Microelectronics, Electronic Components and Materials, Vol. 48, 2018, No. 3, pp. 161–171.

[16] Kamaraj, A.—Marichamy, P.: Design of Fault-Tolerant Reversible Vedic Multiplier in Quantum Cellular Automata. Journal of the National Science Foundation of Sri Lanka, Vol. 47, 2020, No. 4, pp. 371–382, doi: 10.4038/jnsfsr.v47i4.9677.

[17] Thapliyal, H.—Ranganathan, N.: Design of Reversible Sequential Circuits Optimizing Quantum Cost, Delay, and Garbage Outputs. ACM Journal on Emerging Technologies in Computer Systems, Vol. 6, 2010, No. 4, Art. No. 14, pp. 1–31, doi: 10.1145/1877745.1877748.

Kamaraj Arunachalam received his B.E. degree in electronics and communication engineering from Bharathiar University, Coimbatore, Tamil Nadu, India in 2003. He completed his post graduation from Anna University, Chennai in the field of VLSI Design in 2006. Currently he is in the position of Assistant Professor in the Department of Electronics and Communication Engineering, Mepco Schlenk Engineering College, Sivakasi, India. He has completed his Ph.D. at Anna University, Chennai. His research interests include digital circuits and logic design, reversible logic and synthesis and advanced computing techniques. During his 14 years of teaching career, he has published 21 papers in international journals and 23 papers in national and international conferences. He has filed 2 patents and was granted with 1 copyright. He is a member of IETE and ISTE.

Marichamy Perumalsamy obtained his B.E. degree from PSG College of Technology, Coimbatore, M.E. degree from the College of Engineering, Guindy (CEG) – Anna University, Chennai in 1993, and his Ph.D. from the Indian Institute of Technology, Kharagpur, India in 2002. He has more than 34 years of service in teaching. He worked in the National Engineering College, Kovilpatti, India, Nizwa College of Technology, Sultanate of Oman. Currently he is working as Dean in P.S.R. Engineering College, Sivakasi, Tamilnadu, India. He has published more than 28 papers in various international journals. His areas of interest include cellular mobile communication and green networks. He has a Life Time Membership in ISTE.

Kaviyashri K. Ponnusamy was awarded her undergraduate degree in the field of electronics and communication engineering from Kamaraj College of Engineering and Technology, Virudhunagar, Tamil Nadu, India in 2017, and a post graduate degree from Mepco Schlenk Engineering College, Sivakasi in the field of VLSI design in 2019. She is working as Assistant Professor in the Department of Electronics and Communication Engineering, M.A.M. College of Engineering, Trichy, India. Her research interests include digital circuits and logic design. She has published 2 papers in national and international conferences and 1 paper in an international journal.