Computing and Informatics, Vol. 39, 2020, 361–384, doi: 10.31577/cai_2020_3_361

EFFECTIVE UTILIZATION OF SUPERVISED LEARNING TECHNIQUES FOR PROCESS MODEL MATCHING

Khurram Shahzad, Arslaan Mazhar

Punjab University College of Information Technology
University of the Punjab, Lahore, Pakistan
e-mail: {khurram, mscsf14m032}@pucit.edu.pk

Ghulam Mustafa

University of the Punjab, Gujranwala Campus, Pakistan
e-mail: gmustafa@pugc.edu.pk

Faisal Aslam

Punjab University College of Information Technology
University of the Punjab, Lahore, Pakistan
e-mail: faisal.aslam@pucit.edu.pk

Abstract. The recent attempts to use supervised learning techniques for process model matching have yielded below par performance. To address this issue, we have transformed the well-known benchmark correspondences to a readily usable format for supervised learning. Furthermore, we have performed several experiments using eight supervised learning techniques to establish that imbalance in the datasets is the key reason for the abysmal performance. Finally, we have used four data balancing techniques to generate balanced training dataset and verify our solution by repeating the experiments for the four datasets, including the three benchmark datasets. The results show that the proposed approach increases the matching performance significantly.

Keywords: Business process management, process model matching, artificial intelligence, supervised learning techniques, machine learning, data balancing

1 INTRODUCTION

Process models are the conceptual models that represent business operations of an enterprise. These models are widely acknowledged as useful artifacts for documenting software development requirements as well as configuring ERP systems. Process Model Matching (PMM) refers to the identification of the corresponding activities between a pair of process models that represent identical or similar behavior [1]. For a further understanding of PMM problem, consider the excerpt process models of two universities presented in Figure 1. Both process models are composed of a start node represented by a circle, an end node represented by a solid circle, four activities represented by a rectangle with rounded edges, control flow between activities represented by arrow signs, and two gateways represented by a diamond sign having "+" signs. In the figure, the four shaded areas (C1, C2, C3 and C4) represent the four corresponding activity pairs that should be identified by a matching technique based on the similarity of activity labels. While the identification of corresponding activities having identical labels is a trivial task, the real challenge lies in the identification of activities having similar business semantics but different formulation of labels.

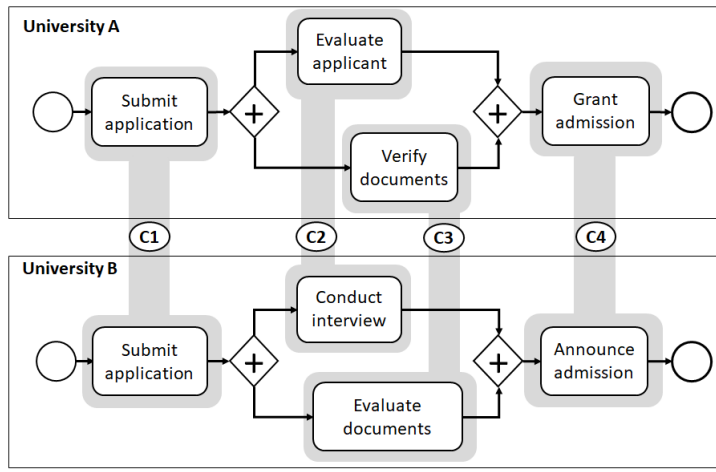


Figure 1. Illustration of the PMM problem

The identification of corresponding activities has several use cases [2]. Firstly, it is a pre-requisite to evaluate if the given process models are similar. Secondly, matching techniques, if embedded to a process model repository, can avoid redundant process models, which may lead to several inconsistencies. Thirdly, it can also play a pivotal role in querying process model repository, as querying involves matching the query model and the models stored in the repository. Finally, harmonizing process model variants is another use of process model matching. Due to these diverse

use cases of PMM, a plethora of techniques have been developed [3, 4]. However, a recent study has highlighted that the F1 score of these techniques vary between 0.45 and 0.67 [5]. This lower value of F1 score highlights the need for developing PMM techniques that can identify corresponding activities with a higher accuracy.

Supervised learning techniques have been widely used in the natural language processing for a variety of text processing tasks, such as word sense disambiguation, text matching, and named entity recognition [6, 7]. It is because the supervised learning techniques use training data to elicit knowledge and subsequently utilize it to predict the solution of a given problem. On the contrary, the unsupervised learning techniques use mathematical models or heuristics to generate a solution of the given problem, without using any insights from the existing solutions. From the text processing literature, it is abundantly established that typically supervised learning techniques have outperformed unsupervised techniques for several text processing tasks [8]. However, the potential of supervised learning techniques is yet to be fully exploited in the context of PMM. For instance, a recent attempt [9] to adapt a supervised learning technique on the PMMC '15 datasets has achieved an F1 score of 0.61 that is substantially less than the maximum F1 score of 0.67, achieved by a traditional unsupervised approach [5].

In this study, we have experimentally established the cause of abysmal performance of supervised learning techniques, and we propose a solution to rectify it. Specifically, we make the following main contributions. Firstly, we have employed a systematic protocol to transform the benchmark correspondences into a readily usable form for the supervised learning techniques. Secondly, we have performed several experiments to establish that the presence of imbalance in the datasets impedes the performance of supervised learning techniques. Precisely, we have used ten different feature measures for training, we tweaked the weights of tokens, and we adjusted the size of training datasets. Finally, to rectify the imbalance problem, we have proposed a data balancing based technique and evaluated its effectiveness for process model matching.

The rest of the paper is organized as follows: Section 2 presents the protocol that we have employed to transform the benchmark datasets. Section 3 investigates the causes of below par performance of the supervised learning techniques. Section 4 presents our proposed solution and the experimental setting that we have used to evaluate the proposed technique. The analysis of the results are presented in Section 5. A brief overview of the related work is presented in Section 6. Finally, in Section 7 we draw conclusions.

2 TRANSFORMATION OF BENCHMARK DATASETS

In this section, we present our first contribution, the transformation of benchmark dataset to make it readily available for supervised learning techniques. Below, we introduce the three widely used datasets, and the protocol that we have used to transform the benchmark datasets.

Source datasets. We conducted a comprehensive search of process model matching literature by querying multiple digital repositories, such as `Springerlink` and `ScienceDirect`, using several keywords. The retrieved items were manually screened to obtain 17 research articles that have experimentally evaluated the effectiveness of PMM techniques. Table 1 provides the list of the studies and the datasets used in these studies for the evaluation of the process model matching techniques. It can be observed from the table that all the existing studies have either used PMMC'15 datasets or their earlier versions for the evaluation. This indicates that any findings that stem from the use of the PMMC'15 datasets are acceptable for the community.

Dataset	**References**
Earlier version of a PMMC'13 dataset	[11, 19, 20]
PMMC'13 datasets	[21, 22]
PMMC'15 datasets	[1, 2, 3, 5, 9, 23, 24, 25, 26, 27, 28, 29]

Table 1. Benchmark datasets used in literature

The PMMC'15 datasets include University Admission (UA), Birth Registration (BR), and Asset Management (AM) collections of 9, 9 and 72 process models, respectively. The models are designed in BPMN, PNML and EPML formats having 289, 238, and 1 993 activities, respectively. Furthermore, each dataset includes 36 process model pairs and benchmark correspondences between activities of each process model pair. The key features of the PMMC'15 datasets that we have used are the following: Firstly, the datasets are publicly available hence the results produced using them are universally verifiable. Secondly, they include real-world process models from three different domains, providing sufficient diversity. Hence, any findings based on these datasets are likely to be applicable to other domains. Lastly, each dataset also includes a collection of gold standard correspondences that can be used as a benchmark for the evaluation of process matching techniques.

Although the collections of process models have sufficient diversity to challenge the capabilities of process matching techniques but the initial screening of the datasets revealed that the benchmark correspondences are not readily usable for the supervised learning techniques. It is due to the following reasons: Firstly, the storage format of each dataset is different. Secondly, the benchmark correspondences are limited to equivalent or optimal equivalent pairs, whereas the sub-optimal equivalent pairs and the pairs in which one label subsumes the other label are not provided. Thirdly, information regarding unequivalent pairs is not explicitly provided.

Transforming the benchmark datasets. In the first transformation step, we wrote a parser that can extract activity labels from BPMN, PNML and EPML formats (the formats in which the three datasets are currently available), and store them in CSV files. Subsequently, we generated a cross-product between all activities of each process model pair which resulted in 36 675, 25 045 and 30 764 activity pairs for the UA, BR and AM datasets, respectively. In the second step, the activity

pairs that were declared equivalent in the PMMC'15 gold standard were marked as equivalent pairs in the cross-product, by parsing the gold standard available in the RDF format. In the third step, we engaged two researchers with expertise in process modeling to evaluate the remaining pairs in the cross-product. Specifically, the experts were explicitly told to mark an activity pair as equivalent even if that activity pair is a sub-optimal or a subsumption pair. Finally, the disagreements between the two researchers were resolved with the help of another expert. Accordingly, for each dataset, we generated a CSV file that contains activity labels, as well as human decision about corresponding pairs. Hence, making the datasets readily available for supervised learning techniques by omitting the hassle of parsing multiple data formats. The detailed specifications of each dataset is presented in Table 2.

	UA	**BR**	**AM**
No. of Model Pairs	36	36	36
No. of Pairs in the Dataset	36 675	25 045	30 764
No. of Positive Examples	232	645	222
No. of Negative Examples	36 443	24 400	30 542
Imbalance Ratio	1:157	1:38	1:138

Table 2. Specification of the human benchmark

3 PROCESS MODEL MATCHING USING SUPERVISED LEARNING

In this section, we present our second contribution, experimentation to identify the cause of the below par performance of supervised learning techniques.

3.1 Supervised Learning Techniques

We have selected eight diverse supervised learning techniques for our experiments as each technique has its own strenghts and weaknesses. These techniques are, Naive Bayes, Simple Logistic, IBK, AdaBoostM1, Decision Table, J48, LMT, and Random Forest. We have selected these techniques due to their effectiveness in text processing tasks.

Among these techniques, Naive Bayes is a robust generative classification algorithm that is less sensitive to noisy data and produces stable predictions. It is widely acknowledged as an effective technique for word disambiguation. Whereas, in Simple Logistic, LogitBoost is used as a base weak learner to fit the logistic models. The repetitions of LogitBoost are cross-validated to produce the optimum results. K-Nearest Neighbor (IBK) selects a feature-space based on the nearest neighbors. J48 is an implementation of a decision tree algorithm to predict class labels. AdaBoostM1 algorithm focuses on the hard-to-learn examples using pseudo-loss function. Decision tables are used as hypothesis-space for supervised learning in the Decision Table algorithm. Logistic Model Trees is the supervised learning

algorithm that combines logistic regression to decision tree learning algorithm to improve results. These techniques reduce both bias and variance due to its constituent methods. Lastly, Random Forest is an ensemble classification technique which uses bagging and decisions trees to predict the class labels.

3.2 Feature Measures

We have selected ten text matching measures that are widely used to compute similarity between a pair of sentences. The reasons for the choice of such a large number of features are twofold: firstly, to supplement the impact of an individual feature on a learning technique. Secondly, to increase the breadth of the knowledge base of a learning technique and thereby offering more opportunities for eliciting the hidden knowledge. A brief overview of these measures are as follows:

Levenshtein distance. A distance based measure that computes distance between two input strings by computing a normalized score of the minimum number of character edit operations required to convert one label into the other. For two labels l_1 and l_2, the Levenshtein distance is computed as follows:

$$edit_{norm}(l_1, l_2) = 1 - \frac{|edit\ \ distance(l_1, l_2)|}{\min(|l_1|, |l_2|)}.$$

Cosine similarity. This similarity measure generates a vector representation of both labels. Subsequently, the similarity is computed by the *cosine* of angle between the two vectors.

$$\cos_{sim}(l_1, l_2) = \frac{\overrightarrow{l_1} \bullet \overrightarrow{l_2}}{\left(|\overrightarrow{l_1}||\overrightarrow{l_2}|\right)}.$$

Euclidean distance. Similar to cosine similarity, it first generates a vector representation of both labels. Subsequently, euclidean distance is the square root of the sum of squared differences between the vectors of two labels. Formally, it is defined as follows:

$$EU_{dis}(l_1, l_2) = \left[\left(\overrightarrow{l_1} - \overrightarrow{l_2}\right) \bullet \left(\overrightarrow{l_1} - \overrightarrow{l_2}\right)\right]^{1/2}.$$

Monge-Elkan. A token based approach in which similarity between two labels is computed by measuring the average of the similarity values between pairs of more similar tokens within label l_1 and l_2 [13]. Formally, it is defined as follows:

$$MonEl_{sim}(l_1, l_2) = \frac{1}{|l_1|} \sum_{i=1}^{|l_1|} \max\{sim(l_{1_i}, l_{2_j})\}_{j=1}^{|l_2|}.$$

Block distance. Block distance is depicted in two dimensions with discrete vectors. It calculates the distance between two data points using a grid-like path. The bock distance between two data points is the aggregation of the differences of their corresponding components. Formally, it is defined as follows:

$$Block_{dist}(l_1, l_2) = \sum_{i=1}^{n} |l_{1i} - l_{2i}|.$$

Jaccard similarity. A set theory based measure that treats each label as a collection of tokens. According to this measure, each label is tokenized into words and represented as a set. Jaccard similarity is then the ratio between the number of common words between the two sets and total number of words in both sets.

$$Jac_{sim}(l_1, l_2) = \frac{|S(l_1) \cap S(l_2)|}{|S(l_1) \cup S(l_2)|}.$$

Jaro-Winkler distance. A type of string edit distance that is faster than the Levenshtein distance as it computes the similarity between two labels by counting the number of matching characters in the strings and transpositions. Formally, it is defined as follows:

$$M_w(l_1, l_2) = M_j + (lp\ (1 - M_j)),$$
$$M_j = \frac{1}{3} + \frac{s}{|l_1|} + \frac{s}{|l_2|} + \frac{s-t}{s}$$

where l is the length of the common prefix, p is the constant scaling factor, and s is the number of matching characters between the two labels. Furthermore, M_j represents Jaro distance, s is the number of matching labels, and t is half the number of transpositions.

TagLink token similarity. It is an adaptive hybrid method of tag-based and link-based similarity in which Tag Commonness (TC) and Link Strength (LS) is dynamically determined. The main idea behind this method is to combine the tag-based and link-based approach to achieve the optimal similarity results. It uses a variation of Jaccard similarity as link-based similarity (δ_{link}) and a variation of tf-idf cosine similarity as a tag-based similarity (δ_{tag}) [33]. Formally, it is defined as follows:

$$TagLink(l_1, l_2) = \frac{TC_{l_1}}{TC_{l_1} + LS_{l_2}} \delta_{tag(l_1,l_2)} + \frac{LS_{l_1}}{TC_{l_1} + LS_{l_2}} \delta_{link(l_1,l_2)}.$$

Matching Coefficient. An elementary vector based approach which counts the number of similar terms on which both vectors are non-zero. This similarity measure is useful if attributes have symmetry in data, i.e., they carry comparable

information. It is sensitive to the variable input and does not generalize very well.

$$MC = \frac{number\ of\ matching\ attributes}{total\ number\ of\ attributes}.$$

Soundex. A phonetic based measure that encodes homophones for the same representation so that they can be matched even when there is a minor difference in spellings. In simple words, the words that rely on similarity of pronunciation rather than the vocabulary are declared as similar.

3.3 Datasets

We have used four datasets for experimentation, three PMMC datasets and another Large (LR) dataset that has been recently developed [31], to demonstrate the external validity of our findings. A key feature of the LR dataset is that it is handcrafted to challenge the abilities of PMM techniques. The LR dataset contains 600 process models from different genres and benchmark correspondences between 406 process models pairs. The dataset contains 89 559 activity pairs, including 6 443 equivalent pairs and 83 115 unequivalent pairs with an imbalance ratio of 1:13. All the four datasets used for experimentation were available in CSV format having four columns, an identifier, a pair of activities, and human decisions.

3.4 Conducting the Experiments

For the experiments each label was tokenized, stop words were removed, and each token was stemmed to generate its corresponding stem. Subsequently, the values of all the 10 features were computed for each activity pair which were given as input features to the eight supervised learning techniques. For each experiment, the corresponding dataset was divided into training and testing datasets. To reduce the bias that might occur due to the choice of training and testing dataset, 10-fold cross-validation was performed for each dataset separately. Note that we have used a widely used features selection technique, Information Gain, to identify the optimal set of features, and subsequently used the optimal set of features for experimentation. However, the effectiveness scores were compromised, therefore, the results presented in this study are generated by all the 10 features.

We have used Precision, Recall and F1 scores to evaluate the effectiveness of supervised learning techniques. In the context of process model matching, Precision (P) refers to the proportion of activity pairs that are declared equivalent by a technique and also marked equivalent in the human benchmark. Recall (R) refers to the proportion of activity pairs that are marked equivalent in the human benchmark and also declared equivalent by a technique, whereas, F1 is the harmonic mean of P and R.

3.5 Results

Table 3 shows the average F1 scores of each supervised learning technique. Note, we have synthesized the results to separately evaluate the effectiveness of each technique for equivalent and unequivalent pairs, separately. From the results we have observed the following:

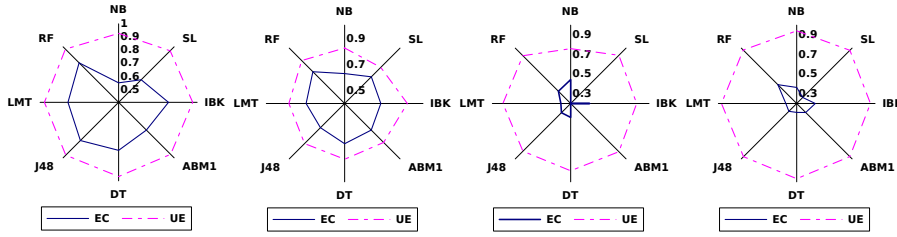
Performance variation between Equivalent (EC) and Unequivalent (UE) pairs. It can be observed from Figure 2 that the F1 scores of EC pairs are significantly less than the corresponding UE pairs. Furthermore, this difference in performance can be observed across all the techniques and for the three PMMC datasets, as well as the additional LR data. Hence, we conclude that the imbalance problem can be generalized as the main cause of the abysmal performance of the supervised learning techniques. We believe that a key reason of this difference in performance is that the unequivalent pairs significantly outnumbered equivalent pairs for all the three datasets. This imbalance in the data limits the learning of each technique and consequently impedes their performance for the equivalent pairs.

Performance variation between techniques. The performance variation between supervised learning techniques is represented by the box plots in Figure 3. It can be observed from the figure that there is a significant variation between the performances of the supervised learning techniques for the equivalent pairs. However, such a variation is not apparent for the unequivalent pairs. These results indicate that the availability of the larger number of unequivalent pairs in the training data provides ample opportunities for *all* the supervised learning techniques to learn and predict the performance of the unequivalent pairs. However, the relatively small number of equivalent pairs in the training data does not provide equal opportunities for all the supervised learning techniques. This is due to two possible reasons, either the available equivalent pairs are not appropriate for an accurate learning, or the available equivalent pairs have contradictions which cannot be resolved due to scarcity of examples. These results confirm our hypothesis that the presence of imbalance in the data impedes the performance of supervised learning techniques.

Performance variation across datasets. It can be observed from Figure 4 that the F1 scores for the unequivalent pairs in all the datasets are comparable. On the contrary, for the equivalent pairs, the F1 scores of AM dataset are significantly less than the other datasets. This indicates that the AM dataset does not have sufficient or appropriate examples to learn and predict the performance of equivalent pairs. It is because the AM dataset contains a large number of process models and activities, whereas the benchmark correspondences merely contain 222 equivalent pairs. Thus, the equivalent pairs of AM dataset do not have sufficient examples to encompass the diversity of their models.

Techniques	UA		BR		AM		LR	
	EC	UE	EC	UE	EC	UE	EC	UE
Naive Bayes	0.549	0.927	0.585	0.846	0.444	0.76	0.364	0.942
Simple Logistic	0.644	0.957	0.505	0.854	0	0.896	0.287	0.968
IBK	0.781	0.966	0.693	0.871	0.393	0.874	0.39	0.953
AdaBoostM1	0.7	0.962	0.592	0.861	0	0.896	0.33	0.967
Decision Table	0.767	0.967	0.606	0.867	0.342	0.887	0.291	0.968
J48	0.813	0.971	0.612	0.857	0.333	0.889	0.313	0.968
LMT	0.787	0.967	0.605	0.864	0.3	0.892	0.311	0.968
Random Forest	0.827	0.975	0.694	0.886	0.377	0.89	0.472	0.972

Table 3. F1 scores of supervised learning techniques (EC = Equivalent pairs and UE = Unequivalent pairs)

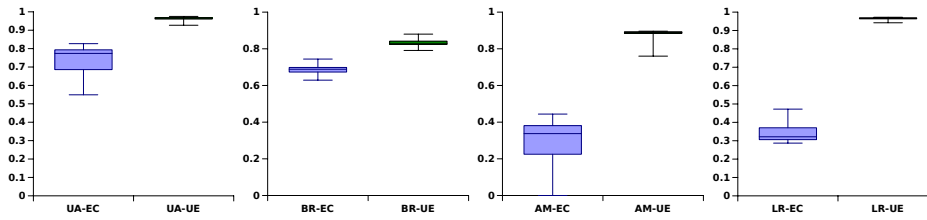


a) UA dataset b) BR dataset c) AM dataset d) LR dataset

Figure 2. Performance variation of F1 scores between equivalent and unequivalent pairs

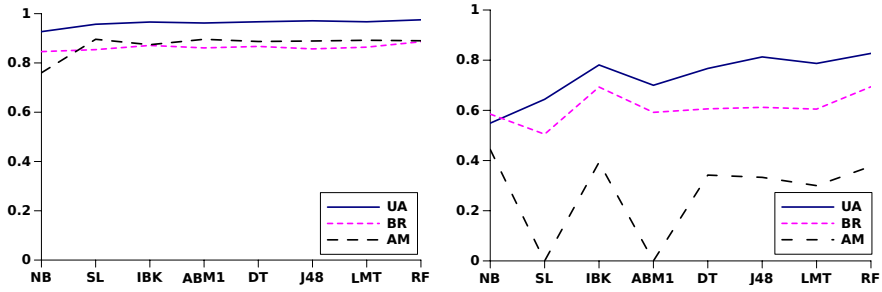
4 THE PROPOSED SOLUTION

In this section, we present our third contribution which is based on the use of data sampling technique to balance the training data and ensure that a supervised learning technique has equal opportunity to elicit the knowledge for equivalent and unequivalent pairs. Listing 1 provides an overview of our proposed solution. It



a) UA dataset b) BR dataset c) AM dataset d) LR dataset

Figure 3. Performance variation of F1 scores between techniques



a) Unequivalent pairs b) Equivalent pairs

Figure 4. Performance variation across datasets of F1 scores between equivalent and unequivalent pairs

includes pseudo-codes for feature extraction, pre-processing of labels, and balancing the training data.

In the listing, Pi is an i^{th} process model, and $L_{Pi,Pj}[\,][\,]$ represents the array of all the activity labels in the two process models, Pi and Pj. Furthermore, $AP_{L_{Pi,Pj}}[\,]$ represents the array of all the activity pairs between the two process models Pi and Pj. Finally, SLT represents a supervised learning technique. In contrast to the existing studies that use term weights as features, we are the first to use a set of similarity scores between activity labels as features. Furthermore, we have also introduced the idea of using sampling techniques to balance the training data in the context of process model matching.

```
float ourAlgo(AP_{L_{Pi,Pj}}[ ]) { // get benchmark decision for each activity
    pair.
   bool bmcDecisions[ ] = getBMC(AP_{L_{Pi,Pj}}[ ]) //10-fold cross-validation
   for (r = 1 to 10) {
      //divide data in two parts, training (TR) and testing (TS) activity
         pairs.
      AP_{L_{Pi,Pj}}[TR][TS] = dataDivider(AP_{L_{Pi,Pj}}[ ], bmcDecisions[ ])
      //check the imbalance ratio a constant.
      if (getImbalanceRatio(bmcDecisions[ ]) < α) { // balance training
         data
         AP_{L_{Pi,Pj}}[TR] = dataBalTech(AP_{L_{Pi,Pj}}[TR])
      }
      //TRAINING
      L_{Pi,Pj}[ ][ ] = preProcessing(AP_{L_{Pi,Pj}}[TR])
      sim[ ][ ] = feature-extraction(L_{Pi,Pj}[ ][ ])
      SLT-Trained = tain&learn(STL, sim[ ][ ])
      //TESTING
      L_{Pi,Pj}[ ][ ] = preProcessing(AP_{L_{Pi,Pj}}[TS])
      sim[ ][ ] = feature-extraction(L_{Pi,Pj}[ ][ ]) // Returns SLT
```

```
        decisions for each pair
      SLTDecisions[ ][ ] =applySLT(SLT-Trained, sim[ ][ ])
           F1Score[r] = computeAccuracy(SLTDecisions[ ][ ], bmcDecisions[
              ])
      }
   avgF1Score = computeAverage (F1Score[ ])
   return avgF1Score
}
/* * Omits trival variations in labels */
L_Pi,Pj[ ][ ] preprocessing(AP_L{Pi,Pj}[ ] ) {
   for (k=1 to length of AP_L{Pi,Pj}[ ]) {
      AP_L{Pi,Pj}[k] = tokenize(AP_L{Pi,Pj}[k])
      AP_L{Pi,Pj}[k] = removeStopWords(AP_L{Pi,Pj}[k]) //coversion to stem
          words
      AP_L{Pi,Pj}[k] = stemming(AP_L{Pi,Pj}[k])
   }
   return L_Pi,Pj[ ][ ]
}
/* * Returns similarity scores of each pair */
sim[ ][ ] feature-extraction(L_Pi,Pj[ ][ ]) {
   for (each metric m) {
      for (k=1 to length of L_Pi,Pj[ ][ ]) {
         sim[k][m]= computeSimilarity(L_Pi,Pj[ ][ ], m)
      }
   }
   return sim[ ][ ]
}
```

Listing 1. Our approach to improve performance of supervised learning techniques

4.1 Data Balancing Techniques

A brief overview of the data balancing techniques are as follows:

Distribution-based balancing. In the distribution-based balancing different probability distributions are learned from imbalanced dataset to form a balanced dataset [15]. We have used Gaussian distribution for balancing the training data using Box and Muller method, as it is widely acknowledged as an sampling technique for adequately approximating the models. In our experiments, it models the univariate relation of class labels with the features.

Spread subsample. It uses random subsamples method to select the *spread* between minority and majority classes [16]. In this method, one can use uniform distribution of samples so that the number of majority class samples are reduced to minority class samples to balance the distribution of imbalanced dataset.

Synthetic minority oversampling (SMOTE). SMOTE is a pseudo oversampling technique in which minority class instances are increased by generating new pseudo instances and thus decrease the spread between minority and majority classes [17]. In particular, firstly, the minority class instances are identified. Subsequently, the neighbors of the minority class instances are selected and new minority class instances are added based on the selected neighbors.

Class balancer. In the class balancer technique, the instances in the dataset are reweighed and as a result each class contains the same total weight [18]. The aggregated weight of all instances is kept constant. The weights of only first batch of instances are altered so that it can be employed with the Filtered Classifier.

4.2 Experimental Setup

For the experiments, we have used the same four datasets that were used in the experiments presented in Section 3.4. Furthermore, the same evaluation measures, preprocessing, similarity scores, features, training-testing ratio, 10-fold cross-validation and the same supervised learning algorithms have been used. However, for conducting the experiments, we have employed four different data balancing techniques to identify a training dataset, whereas the testing dataset remained unchanged.

4.3 Results

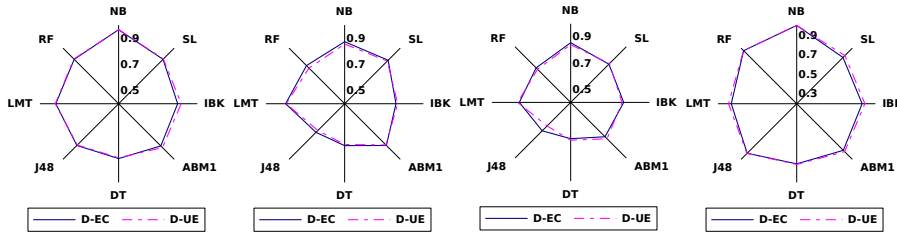
Table 4 shows the average F1 score of 10-fold cross-validation for each supervised learning technique. Similar to the results of the previous experiments, the results of equivalent and unequivalent pairs are separated to highlight the differences between their scores. Note, we generated P, R and F1 scores separately for 12 data subsets, which are produced by applying 4 data balancing techniques on UA, BR and AM datasets. However, for brevity only the F1 scores are presented in the Table 4. From the results, we have observed the following:

Reduced performance variation between equivalent and unequivalent pairs. It can be observed from Figure 5, that after data balancing the F1 scores for equivalent pairs become comparable with the unequivalent pairs. Additionally, from the comparison of Figure 2 and 5, it can be observed that the variation between the F1 scores of equivalent and unequivalent pairs is significantly reduced. This indicates that all the balancing techniques choose appropriate and sufficient examples for equivalent as well as unequivalent pairs. It further indicates that the chosen examples are effective for learning and prediction of equivalent and unequivalent pairs.

Reduced performance variation between techniques. From the comparison of Figure 3 and 6, it can be observed that the sizes of box plot quartiles for

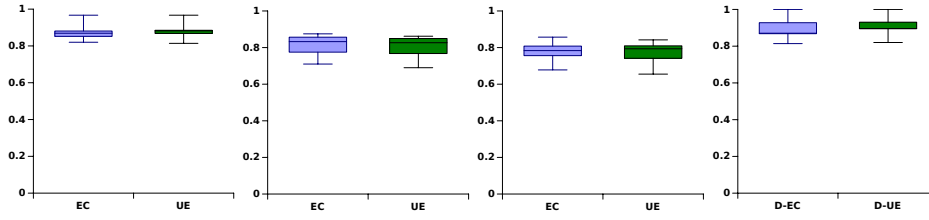
Techniques	Distribution Base		Undersampling		SMOTE		Class Balancer	
	EC	UE	EC	UE	EC	UE	EC	UE
UA Dataset								
Naive Bayes	0.967	0.967	0.682	0.766	0.689	0.733	0.686	0.771
Simple Logistic	0.881	0.885	0.861	0.861	0.867	0.831	0.833	0.838
IBK	0.852	0.879	0.908	0.909	0.977	0.97	0.903	0.911
AdaBoostM1	0.857	0.875	0.817	0.853	0.841	0.763	0.82	0.857
Decision Table	0.82	0.814	0.841	0.873	0.938	0.922	0.856	0.866
J48	0.852	0.847	0.909	0.908	0.965	0.956	0.912	0.916
LMT	0.881	0.885	0.9	0.897	0.97	0.962	0.902	0.91
Random Forest	0.881	0.885	0.923	0.928	0.98	0.975	0.916	0.922
BR Dataset								
Naive Bayes	0.875	0.857	0.667	0.752	0.653	0.78	0.662	0.75
Simple Logistic	0.871	0.862	0.631	0.718	0.615	0.765	0.64	0.722
IBK	0.793	0.806	0.821	0.812	0.872	0.887	0.799	0.818
AdaBoostM1	0.852	0.847	0.65	0.741	0.646	0.785	0.664	0.757
Decision Table	0.721	0.712	0.677	0.759	0.673	0.785	0.682	0.762
J48	0.71	0.69	0.679	0.745	0.79	0.823	0.779	0.781
LMT	0.852	0.847	0.687	0.695	0.789	0.827	0.767	0.777
Random Forest	0.813	0.786	0.817	0.816	0.849	0.878	0.786	0.813
AM Dataset								
Naive Bayes	0.857	0.842	0.697	0.68	0.673	0.646	0.687	0.639
Simple Logistic	0.814	0.82	0.709	0.687	0.699	0.678	0.689	0.658
IBK	0.806	0.793	0.679	0.658	0.846	0.847	0.675	0.723
AdaBoostM1	0.772	0.794	0.719	0.69	0.717	0.691	0.727	0.689
Decision Table	0.678	0.689	0.73	0.683	0.748	0.72	0.741	0.689
J48	0.708	0.655	0.768	0.709	0.77	0.708	0.743	0.676
LMT	0.793	0.806	0.746	0.7	0.784	0.749	0.736	0.684
Random Forest	0.774	0.759	0.664	0.686	0.837	0.851	0.717	0.748
LR Dataset								
Naive Bayes	1	1	0.587	0.744	0.564	0.812	0.59	0.743
Simple Logistic	0.868	0.896	0.638	0.729	0.683	0.921	0.64	0.733
IBK	0.868	0.896	0.655	0.652	0.821	0.932	0.539	0.743
AdaBoostM1	0.873	0.892	0.595	0.74	0.565	0.813	0.572	0.740
Decision Table	0.814	0.82	0.64	0.732	0.654	0.882	0.629	0.73
J48	0.915	0.918	0.633	0.732	0.898	0.94	0.612	0.743
LMT	0.868	0.896	0.638	0.73	0.73	0.812	0.612	0.752
Random Forest	0.966	0.968	0.69	0.735	0.843	0.883	0.673	0.732

Table 4. Result of supervised learning techniques using balanced datasets



a) UA dataset b) BR dataset c) AM dataset d) LR dataset

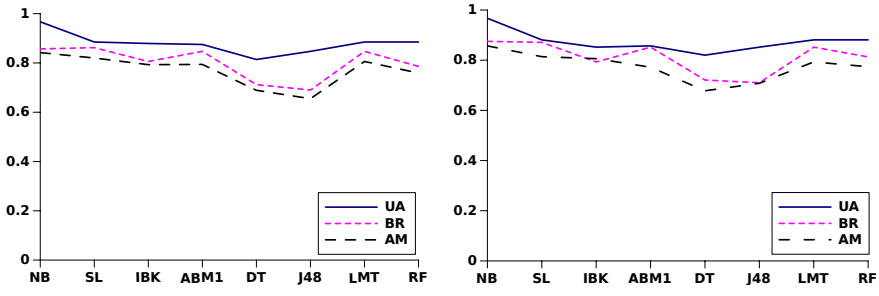
Figure 5. Performance variation of F1 scores between equivalent and unequivalent pairs after data balancing



a) UA dataset b) BR dataset c) AM dataset d) LR dataset

Figure 6. Performance variation of F1 scores between techniques after data balancing

equivalent pairs are decreased for UA and AM datasets, representing that the variation between the performance of supervised learning techniques is reduced for these datasets. It implies that the data balancing has provided equal opportunity for all the supervised learning techniques for UA and AM datasets. On the contrary, the size of the quartiles for BR dataset has increased slightly



a) Unequivalent pairs b) Equivalent pairs

Figure 7. Performance variation across datasets of equivalent and unequivalent pairs after applying data balancing

after data balancing, indicating that the variation between performance of the supervised learning techniques has increased.

Reduced performance variation across datasets. It can be observed from Figure 7 that for the equivalent as well as the unequivalent pairs there is no significant gap between the performances of the supervised learning techniques across the three datasets. This indicates that data balancing has provided better opportunities to all the techniques across the three datasets.

5 ANALYSIS OF THE RESULTS

To evaluate the significance of performance gain that was achieved due to the proposed approach, we have applied Friedman ANOVA test between balanced and imbalanced datasets. The test is applied to the F1 scores of the 10-folds by setting significance level to 0.05. The results of Distribution based balancing technique are presented in Table 5. From the results presented in Table 5 we have observed the following:

Significant performance gain: The increase in the performance of the supervised learning techniques is statistically significant for the equivalent pairs. This observation is valid for all the supervised learning techniques and across the three datasets.

Insignificant performance reduction for unequivalent pairs: For unequivalent pairs, the reduction in the performance is statistically insignificant in majority of the cases. Furthermore, in a few cases the performance even improved significantly.

Based on the above observations, we conclude that the use of data balancing in supervised learning techniques enhances the efficiency of process model matching, whenever the imbalance is large. However, the balance techniques may not be equally effective when the imbalance ratio is small.

6 RELATED WORK

Process model matching was initially considered a rudimentary problem for computing similarity between process models, querying process model repositories, harmonization of process models, detection of process clones [32, 34], etc. However, recent studies [2, 3, 4] have recognized the importance of process model matching beyond its traditional usage. Consequently, a plethora of process model matching techniques have been developed. We have conducted a comprehensive survey of process model matching techniques by employing a snowballing approach. More specifically, we started with a process model matching survey [5] and performed forward and backward tracing to identify the studies that focus on identifying corresponding activities between a pair of process models. Accordingly, we identified

Technique	UA Dataset		BR Dataset		AM Dataset	
	EC	**UE**	**EC**	**UE**	**EC**	**UE**
Naive Bayes	S+	S+	S+	S+	S+	S+
Simple Logistic	S+	NS	S+	NS	S+	NS
IBK	S+	NS	S+	NS	S+	NS
AdaBoostM1	S+	S-	S+	S-	S+	S-
Decision Table	NS	NS	S+	NS	S+	NS
J48	S+	S-	S+	S-	S+	S-
LMT	S+	NS	S+	NS	S+	NS
Random Forest	S+	S-	S+	S-	S+	S-

Table 5. Friedman-ANOVA test between imbalanced and distribution based balanced datasets. S represents significant, NS insignificant, where + and − represent increase and decrease in performance, respectively.

domains specific, as well as generic studies. However, for brevity, we have only discussed generic studies. Our comprehensive examination of techniques revealed that the matching techniques can be subdivided into two broad categories, supervised and unsupervised techniques.

Unsupervised Techniques. These techniques use text matching measures to identify corresponding activities between a pair of process models. Typically, these techniques are composed of two phases [9]. The first phase computes similarity score between activities of process models, whereas the second phase converts the similarity score into a binary decision of corresponding activities or not [5]. These techniques are further divided into syntactic and semantic measures [5]. A brief overview of these techniques are as follows:

Syntactic measures: The measures in this category merely rely on the surface form of the words that constitute the labels of the participating activities. That is, these measures compute the similarity between a given pair of activities by tokenizing labels into words, and subsequently comparing the words by using string comparison operations [35]. Typically, distance-based measures, such as Edit-distance, Levenshtein distance, Hamming distance, and Jensen-Shannon distance, have been used for the comparison. These measures compute the similarity between two words by counting the minimum number of string edit operations (insertion, deletion, and update) required to convert one word into the other word. A lower value of edit distance represents higher similarity between the labels and vice versa.

In addition to the distance-based measures, Dice similarity and Cosine similarity are also used for syntactic comparison. Dice similarity is the ratio between the number of shared words between two activities and the total number of words in the two activities [36]. On the contrary, Cosine similarity transforms each word or a label into a vector and computes the similarity as cosine of angle between

the activity vectors [19]. Furthermore, other syntactic similarity measures, such as Jaccard similarity and Longest Common Subsequence, have been used for process matching.

The syntactic measures yield high accuracy in case the vocabulary of participating activity labels is comparable, however there are at least two cases in which these measures do not yield high accuracy:

1. the considered activities are composed of unrelated words having similar spellings, such as "give contract" and "live contact",
2. the activities are composed of words with same meaning but different vocabulary, such as "evaluate applicant" and "assess candidate".

Semantic measures. These measures address the limitations of the syntactic measures as they take into account the meanings or relatedness of the considered words. To that end, this technique relies on a large lexical English database, called WordNet [37]. Specifically, the participating labels are tokenized into words and pre-processed to generate root form of each word. Subsequently, the similarity between words is computed by using the *similarity* or *relatedness* of words.

The similarity based measures rely on the similarity between two words by considering their synonyms. The examples of the similarity based techniques are Static weighted word comparison [38], Dice with synonyms, and intersection of synonyms. These techniques yield higher matching accuracy between the labels in which a concept is represented by different words.

In contrast to the similarity based measures, a relatedness based measure takes into consideration the co-relatedness of words, represented by `is-a` relationship between the concepts in the WordNet topology. That is, the words having shorter path between them are considered more similar that the ones having longer path. Lin, Lesk, Wu & Palmer, Leacock, and Jiang similarity are the relatedness based semantic measures [39].

Supervised Techniques. Recent studies have attempted to increase the accuracy of process model matching using supervised learning techniques [9, 3, 40]. The F1 scores achieved by the supervised learning technique for the three PMMC'15 datasets are 0.54, 0.38 and 0.61, which are slightly higher than the F1 scores achieved by the matchers participating in the latest edition of the PMM contest in 2015. Furthermore, a state-of-the-art approach has proposed to combine the strengths of individual matchers using an ensemble of multiple matchers [3] to improve the accuracy of PMM. The most recent study [40] has proposed a word-embeddings based approach to increase the accuracy of process matching to achieve an F1 score of 0.84, 0.72 and 0.91. A key limitation of the proposed approach is that its analysis is limited to a unified F1 score, without distinguishing between equivalent and unequivalent pairs.

7 CONCLUSION

Due to the growing interest in process model matching, a plethora of unsupervised learning techniques have been developed. Recent attempts have been made to introduce supervised learning for process model matching, without achieving a significantly higher accuracy than the traditional unsupervised techniques. This is an anomaly, because the supervised learning techniques are proven to significantly outperform unsupervised techniques for a variety of text processing tasks. Therefore, the aim of this paper is to investigate the cause behind the abysmal performance of the supervised learning techniques in process model matching and suggest a solution to improve their performance.

To this end, in this paper, we have made three main contributions. Firstly, we have transformed the existing benchmark correspondences into a readily usable form for the supervised learning techniques. Secondly, we have conducted a series of experiments using eight state-of-the-art supervised learning techniques and synthesized the results to establish that the presence of imbalance in the datasets adversely affects the matching results. Thirdly, we applied four different data balancing techniques to achieve groundbreaking accuracy in the process model matching. That is, our proposed solution achieved a maximum F1 score of 0.98, whereas the plethora of existing techniques for process model matching (including both supervised and unsupervised techniques) were able to achieve a maximum F1 score of merely 0.67. Furthermore, even the average F1 score of 0.79 achieved by our solution is higher than the maximum F1 score of 0.67 achieved by all the existing techniques. In the future, we plan to use the state-of-the-art deep learning techniques for process model matching.

REFERENCES

[1] Kuss, E.—Leopold, H.—van der Aa, H.—Stuckenschmidt, H.—Reijers, H. A.: Probabilistic Evaluation of Process Model Matching Techniques. In: Comyn-Wattiau, I., Tanaka, K., Song, I. Y., Yamamoto, S., Saeki, M. (Eds.): Conceptual Modeling (ER 2016). Springer, Cham, Lecture Notes in Computer Science, Vol. 9974, 2016, pp. 279–292, doi: 10.1007/978-3-319-46397-1_22.

[2] Antunes, G.—Bakhshandeh, M.—Borbinha, J.—Cardoso, J.—Dadashnia, S. et al.: The Process Model Matching Contest 2015. In: Kolb, J. et al. (Eds.): Enterprise Modelling and Information Systems Architectures. Gesellschaft für Informatik, Bonn, Lecture Notes in Informatics, Vol. 248, 2015, pp. 127–155.

[3] Meilicke, C.—Leopold, H.—Kuss, E.—Stuckenschmidt, H.—Reijers, H. A.: Overcoming Individual Process Model Matcher Weaknesses Using Ensemble Matching. Decision Support Systems, Vol. 100, 2017, pp. 15–26, doi: 10.1016/j.dss.2017.02.013.

[4] Kuss, E.—Leopold, H.—Meilicke, C.—Stuckenschmidt, H.: Ranking-Based Evaluation of Process Model Matching. In: Panetto, H. et al. (Eds.): On the Move to

Meaningful Internet Systems (OTM 2017). Springer, Cham, Lecture Notes in Computer Science, Vol. 10573, 2017, pp. 298–305, doi: 10.1007/978-3-319-69462-7_19.

[5] Jabeen, F.—Leopold, H.—Reijers, H. A.: How to Make Process Model Matching Work Better? An Analysis of Current Similarity Measures. In: Abramowicz, W. (Ed.): Business Information Systems (BIS 2017). Springer, Cham, Lecture Notes in Business Information Processing, Vol. 288, 2017, pp. 181–193, doi: 10.1007/978-3-319-59336-4_13.

[6] Malik, M. K.: Urdu Named Entity Recognition and Classification System Using Artificial Neural Network. ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), Vol. 17, 2017, No. 1, Art. No. 2, 13 pp., doi: 10.1145/3129290.

[7] Yahaya, M. F.—Rahman, N. A.—Bakar, Z. A.—Hasmy, H.: Evaluation on Knowledge Extraction and Machine Learning in Resolving Malay Word Ambiguity. Journal of Fundamental and Applied Sciences, Vol. 9, 2017, No. 5S, pp. 115–130, doi: 10.4314/jfas.v9i5s.10.

[8] Pang, B.—Lee, L.: Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval. Vol. 2, 2008, No. 1–2, pp. 1–135, doi: 10.1561/1500000011.

[9] Sonntag, A.—Hake, P.—Fettke, P.—Loos, P.: An Approach for Semantic Business Process Model Matching Using Supervised Machine Learning. 24th European Conference on Information Systems (ECIS), Istanbul, Turkey, 2016, Research-in-Progress Papers, Art. No. 47.

[10] Cardoso, J.—Bakhshandeh, M.—Faria, D.—Pesquita, C.—Borbinha, J.: Ontology-Based Approach for Heterogeneity Analysis of EA Models. In: Dumas, M., Fantinato, M. (Eds.): Business Process Management Workshops (BPM 2016). Springer, Cham, Lecture Notes in Business Information Processing, Vol. 281, 2016, pp. 131–142, doi: 10.1007/978-3-319-58457-7_10.

[11] Klinkmüller, C.—Weber, I.—Mendling, J.—Leopold, H.—Ludwig, A.: Increasing Recall of Process Model Matching by Improved Activity Label Matching. In: Daniel, F., Wang, J., Weber, B. (Eds.): Business Process Management. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 8094, 2013, pp. 211–218, doi: 10.1007/978-3-642-40176-3_17.

[12] Allweyer, T.: BPMN 2.0: Introduction to the Standard for Business Process Modeling. BoD – Books on Demand, 2016.

[13] del Pilar Angeles, M.—Espino-Gamez, A.: Comparison of Methods Hamming Distance, Jaro, and Monge-Elkan. The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications (DBKDA 2015), Rome, Italy, 2015, pp. 63–69.

[14] Büttcher, S.—Clarke, C. L. A.—Cormack, G. V.: Information Retrieval: Implementing and Evaluating Search Engines. The MIT Press, 2016.

[15] Bermejo, P.—Gámez, J. A.—Puerta, J. M.: Improving the Performance of Naive Bayes Multinomial in E-Mail Foldering by Introducing Distribution-Based Balance of Datasets. Expert Systems with Applications, Vol. 38, 2011, No. 3, pp. 2072–2080, doi: 10.1016/j.eswa.2010.07.146.

[16] Tan, M.—Tan, L.—Dara, S.—Mayeux, C.: Online Defect Prediction for Imbalanced Data. Proceedings of the 37th International Conference on Software Engineering, Vol. 2, IEEE Press, 2015, pp. 99–108, doi: 10.1109/ICSE.2015.139.

[17] Chawla, N.—Bowyer, K. W.—Hall, L. O.—Kegelmeyer, W. P.: SMOTE: Synthetic Minority Over-Sampling Technique. Journal of Artificial Intelligence Research, Vol. 16, 2002, pp. 321–357, doi: 10.1613/jair.953.

[18] Alinejad-Rokny, H.—Sadroddiny, E.—Scaria, V.: Machine Learning and Data Mining Techniques for Medical Complex Data Analysis (Editorial). Neurocomputing, Vol. 276, 2017, p. 1, doi: 10.1016/j.neucom.2017.09.027.

[19] Weidlich, M.—Dijkman, R.—Mendling, J.: The ICoP Framework: Identification of Correspondences Between Process Models. In: Pernici, B. (Ed.): Advanced Information Systems Engineering (CAiSE 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6051, 2010, pp. 483–498, doi: 10.1007/978-3-642-13094-6_37.

[20] Leopold, H.—Niepert, M.—Weidlich, M.—Mendling, J.—Dijkman, R.—Stuckenschmidt, H.: Probabilistic Optimization of Semantic Process Model Matching. In: Barros, A., Gal, A., Kindler, E. (Eds.): Business Process Management (BPM 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7481, 2012, pp. 319–334, doi: 10.1007/978-3-642-32885-5_25.

[21] Cayoglu, U.—Dijkman, R.—Dumas, M.—Fettke, P.—Garcia-Banuelos, L. et al.: Report: The Process Model Matching Contest 2013. In: Lohmann, N., Song, M., Wohed, P. (Eds.): Business Process Management Workshops (BPM 2013). Springer, Cham, Lecture Notes in Business Information Processing, Vol. 171, 2013, pp. 442–463, doi: 10.1007/978-3-319-06257-0_35.

[22] Klinkmüller, C.—Leopold, H.—Weber, I.—Mendling, J.—Ludwig, A.: Listen to Me: Improving Process Model Matching Through User Feedback. In: Sadiq, S., Soffer, P., Völzer, H. (Eds.): Business Process Management (BPM 2014). Springer, Cham, Lecture Notes in Computer Science, Vol. 8659, 2014, pp. 84–100, doi: 10.1007/978-3-319-10172-9_6.

[23] Niesen, T.—Dadashnia, S.—Fettke, P.—Loos, P.: A Vector Space Approach to Process Model Matching Using Insights from Natural Language Processing. Multikonferenz Wirtschaftsinformatik (MKWI), Illmenau, Germany, 2016, pp. 93–104.

[24] Schoknecht, A.—Fischer, N.—Oberweis, A.: Process Model Search Using Latent Semantic Analysis. In: Dumas, M., Fantinato, M. (Eds.): Business Process Management Workshops (BPM 2016). Springer, Cham, Lecture Notes in Business Information Processing, Vol. 281, 2016, pp. 283–295, doi: 10.1007/978-3-319-58457-7_21.

[25] Leopold, H.—van der Aa, H.—Pittke, F.—Raffel, M.—Mendling, J.—Reijers, H. A.: Searching Textual and Model-Based Process Descriptions Based on a Unified Data Format. Software and Systems Modeling, Vol. 18, 2019, pp. 1179–1194, doi: 10.1007/s10270-017-0649-y.

[26] Klinkmüller, C.: Adaptive Process Model Matching: Improving the Effectiveness of Label-Based Matching Through Automated Configuration and Expert Feedback. Ph.D. Dissertation, Macquarie University, Sydney, Australia, 2017.

[27] Kuss, E.—Stuckenschmidt, H.: Automatic Classification to Matching Patterns for Process Model Matching Evaluation. In: Cabanillas, C., España, S., Farshidi, S. (Eds.): Proceedings of the ER Forum 2017 and the ER 2017 Demo Track. 36th International Conference on Conceptual Modeling (ER 2017), Valencia, Spain, 2017. CEUR Workshop Proceedings, Vol. 1979, 2017, pp. 306–319.

[28] Ali, M.—Shahzad, K.: Enhanced Benchmark Datasets for a Comprehensive Evaluation of Process Model Matching Techniques. In: Pergl, R., Babkin, E., Lock, R., Malyzhenkov, P., Merunka, V. (Eds.): Enterprise and Organizational Modeling and Simulation (EOMAS 2018). Springer, Cham, Lecture Notes in Business Information Processing, Vol. 332, 2018, pp. 107–122, doi: 10.1007/978-3-030-00787-4_8.

[29] Kuss, E.—Leopold, H.—van der Aa, H.—Stuckenschmidt, H.—Reijers, H. A.: A Probabilistic Evaluation Procedure for Process Model Matching Techniques. Data and Knowledge Engineering, Vol. 117, 2018, pp. 393–406, doi: 10.1016/j.datak.2018.04.008.

[30] Rana, M.—Shahzad, K.—Nawab, R. M. A.—Leopold, H.—Babar, U.: A Textual Description Based Approach to Process Matching. In: Horkoff, J., Jeusfeld, M., Persson, A. (Eds.): The Practice of Enterprise Modeling (PoEM 2016). Springer, Cham, Lecture Notes in Business Information Processing, Vol. 267, 2016, pp. 194–208, doi: 10.1007/978-3-319-48393-1_14.

[31] Shahzad, K.—Nawab, R. M. A.—Abid, A.—Sharif, K.—Ali, F.—Aslam, F.—Mazhar, A.: A Process Model Collection and Gold Standard Correspondences for Process Model Matching. IEEE Access, Vol. 7, 2019, pp. 30708–30723, doi: 10.1109/ACCESS.2019.2900174.

[32] Becker, M.—Laue, R.: A Comparative Survey of Business Process Similarity Measures. Computers in Industry, Vol. 63, 2012, No. 2, pp. 148–167, doi: 10.1016/j.compind.2011.11.003.

[33] Phan, N. H.—Hoang, V. D. T.—Shin, H.: Adaptive Combination of Tag and Link-Based User Similarity in Flickr. Proceedings of the 18th ACM International Conference on Multimedia (MM '10), ACM, 2010, pp. 675–678, doi: 10.1145/1873951.1874049.

[34] Ekanayake, C. C.—Dumas, M.—García-Bañuelos, L.—La Rosa, M.—ter Hofstede, A. H.: Approximate Clone Detection in Repositories of Business Process Models. In: Barros, A., Gal, A., Kindler, E. (Eds.): Business Process Management (BPM 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7481, 2012, pp. 302–318, doi: 10.1007/978-3-642-32885-5_24.

[35] Dijkman, R.—Dumas, M.—García-Bañuelos, L.: Graph Matching Algorithms for Business Process Model Similarity Search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H. A. (Eds.): Business Process Management (BPM 2009). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5701, 2009, pp. 48–63, doi: 10.1007/978-3-642-03848-8_5.

[36] Jin, T.—Wang, J.—La Rosa, M.—ter Hofstede, A.—Wen, L.: Efficient Querying of Large Process Model Repositories. Computers in Industry, Vol. 64, 2013, No. 1, pp. 41–49, doi: 10.1016/j.compind.2012.09.008.

[37] Miller, G. A.: WordNet: A Lexical Database for English. Communications of the ACM, Vol. 38, 1995, No. 11, pp. 39–41, doi: 10.1145/219717.219748.

[38] Dijkman, R.—Dumas, M.—van Dongen, B.—Käärik, R.—Mendling, J.: Similarity of Business Process Models: Metrics and Evaluation. Information Systems, Vol. 36, 2011, No. 2, pp. 498–516, doi: 10.1016/j.is.2010.09.006.

[39] Shahzad, K.—Pervaz, I.—Nawab, A.: WordNet-Based Semantic Similarity Measures for Process Model Matching. In: Zdravkovic, J., Grabis, J., Nurcan, S., Stirna, J. (Eds.): Joint Proceedings of the BIR 2018 Short Papers, Workshops and Doctoral Consortium, 17th International Conference Perspectives in Business Informatics Research (BIR 2018), Stockholm, Sweden, 2018. CEUR Workshops Proceedings, Vol. 2218, 2018, pp. 33–44.

[40] Shahzad, K.—Kanwal, S.—Malik, K.—Aslam, F.—Ali, M.: A Word-Embedding-Based Approach for Accurate Identification of Corresponding Activities. Computers and Electrical Engineering, Vol. 78, 2019, No. 1, pp. 218–229, doi: 10.1016/j.compeleceng.2019.07.011.

Khurram Shahzad is Tenure Track Assistant Professor at the Punjab University College of Information Technology (PUCIT), University of the Punjab, Lahore. He obtained his Masters and Ph.D. from KTH – Royal Institute of Technology, Stockholm. He has been associated with Information Systems Groups at Technical University Eindhoven, Eindhoven, and University of Fribourg, Fribourg. He has published more than 35 papers in international conferences and journals.

Arslaan Mazhar received his Bachelor's degree in software engineering from the Mirpur University of Science and Technology, Mirpur, Pakistan, and his M.Phil. degree in computer science from Punjab University College of Information Technology, University of the Punjab, Lahore.

Ghulam Mustafa received his M.Sc. degree in computer science from COMSATS Institute of Information Technology (CIIT), Lahore, Pakistan, in 2007 and his Ph.D. degree in computer science and technology from Beijing Institute of Technology (BIT), Beijing, China in 2015. He is currently Assistant Professor at the Department of Information Technology, The University of the Punjab, Gujranwala Campus, Pakistan. His research interests include artificial intelligence, recommender systems and machine learning.

Faisal Aslam received his Ph.D. from University of Freiburg, Germany and Post Doctorate from TU Delft, The Netherlands. He was also Research Fellow at Lund University, Sweden. Currently, he is Assistant Professor at the University of the Punjab. He has published in reputed journals and conferences.

Computing and Informatics, Vol. 39, 2020, 385–409, doi: 10.31577/cai_2020_3_385

PERCEPTUAL QUALITY ASSESSMENT OF DIGITAL IMAGES USING DEEP FEATURES

Nisar Ahmed

Department of Computer Engineering
University of Engineering and Technology, Lahore
Pakistan
e-mail: nisarahmedrana@yahoo.com

Hafiz Muhammad Shahzad Asif

Department of Computer Science
University of Engineering and Technology, Lahore
Pakistan
e-mail: shehzad@uet.edu.pk

Abstract. Perceptual quality assessment is a tough task especially in the absence of reference information. No-reference image quality assessment is more challenging than full-reference or reduced reference methods, as the system has to model the different image distortions in the form of a quality score. Most of the approaches are based on handcrafted features which are based on natural scene statistics and are specific to some distortion types. These approaches provide high correlation with human opinion score for datasets containing specific distortions, but they fail to generalize well in scenarios were multiple distortions or real-time distortions are present in images. Deep learning algorithms, on the other hand, demonstrated their abilities to learn expert features with better discriminatory power for various classification and regression tasks. It is a big challenge to use those deep learning methods for image quality assessment as the image datasets with human opinion score are very small and cannot be used effectively to train a deep learning algorithm. We experimented with activations of different deep layers of thirteen pre-trained models and checked for their suitability for the task of no-reference quality assessment. Fine-tuning of these models on quality assessment datasets provided even better performance. A Gaussian process regression model is trained on these activations to perform the quality assessment and it provided state-of-the-art performance. Cross-dataset validation demonstrated its performance further and also provided further prospects of research in this direction.

Keywords: Image quality, image quality assessment, IQA, deep features, perceptual quality assessment

Mathematics Subject Classification 2010: 60Gxx

1 INTRODUCTION

Assessment of perceptual quality of digital images can be very useful in several image processing applications. Image quality assessment algorithms can be used to monitor the video or image quality to optimize the parameters of image-processing algorithms. Such system can be used to adjust compression ratio, amount of color saturation, contrast adjustment, etc. It can also be used to evaluate the performance of image acquisition hardware and amount of perceptual distortions occurring during transmission. Usually the above applications are based on full-reference methods which require the original image to compare the extracted features with distorted image for assessment of amount of distortion, but this approach has practical limitations. Full-reference methods can be used to assess the compression performance, but they cannot be used to assess the perceptual distortion in transmitted video when the original one is not available (e.g. broadcasting) or in case of evaluation of image acquisition/enhancement. Moreover, full-reference approaches measure the amount of change in original and distorted image but in case of image enhancement applications such as contrast enhancement the perceptual quality of reproduced image is better than the original image and full-reference approaches fail to provide quality score for these applications.

No-reference image quality assessment is therefore crucial for several image processing systems for evaluation of perceptual image quality. These approaches do not require any reference image for comparison but they extract or learn discriminatory features from images which can be used to assess the perceptual quality. On the contrary, due to lack of information, it is harder for no-reference image quality assessment algorithms to assess the perceptual quality better than full-reference approaches. It is therefore more difficult for no-reference approaches to adapt to the behavior of human visual system and result in decreased prediction performance in terms of correlation with Mean Opinion Score (MOS).

No-reference quality assessment is usually performed by extracting handcrafted features such as natural scene statistics and then training a regression algorithm to obtain the quality score. Many no-reference image quality assessment algorithms are proposed in the literature and development of a robust quality assessment algorithm is dependent on quality discrimination ability of the features. Mittal et al. [1], Liu et al. [2] and Sazzad et al. [3] extracted features in the spatial domain. Saad et al. [4], Ma et al. [5] and Liu et al. [6] worked on transform domain features. He at el. [7] and Chang et al. [8] used sparse representation for image quality assessment. These approaches can predict perceptual quality in high correlation to the human judgments.

Deep features, on the other hand, are the activations of convolutional neural networks which are extracted to perform different classification and regression tasks [9, 10, 7, 11]. These features demonstrated very powerful capabilities for image quality assessment task as well [12, 13, 14, 15]. There are two approaches to extract deep features, one is to use the pre-trained CNN model and extract deep features [9, 15] and the other is to fine-tune the model on your problem set and then perform the feature extraction [10]. The second approach is required on the problem of image quality as it is of different nature than the original dataset (ImageNet) which is used in the pre-trained model. Some researchers designed their own architecture [16, 17, 18] or used a previously designed architecture to train from scratch for feature extraction. The deep feature based approaches are much better at predicting perceptual image quality than the handcrafted feature based. However, there is no clarity as how to obtain most representative feature set for perceptual image quality assessment.

In this work, we have performed an analysis of different deep features to identify the most representative feature set for image quality assessment. Our contributions are twofold, the first involves identification of most suitable way of deep feature extraction and the second is construction of a quality prediction model by using Gaussian process regression (GPR). The identification of deep features extraction method is performed by first selecting thirteen popular CNN architectures, pre-trained on ImageNet, and performing feature extraction at different bottleneck layers. Eight of these pre-trained models are fine-tuned on image quality database and then deep features are extracted in a similar manner. The best performing feature set is used to train a GPR as it has been demonstrated that the GPR is the most suitable regression algorithm for the task of image quality assessment. The specific contributions are highlighted below:

- We have provided a comparison of deep features performance for image quality using several popular pre-trained models with and without fine-tuning.
- We have extracted deep features at several bottleneck points and provided three best points to extract features in these architectures for image quality.
- We have highlighted and used NASNet after fine-tunning for feature extraction as it provided most quality aware features.
- We have proposed a Gaussian process regression based model trained using deep features to obtain state-of-the-art performance on several benchmark databases.

2 RELATED WORK

Bosse et al. [19] presented a deep neural network based quality assessment approach. They follow configuration of a Siamese network in which the differences of extracted features for original and distorted images are taken and features are fused to perform regression with fully connected neural network. The whole process is applied in patch-wise fashion and weighted averaging is used for final quality score. As their

approach requires both pristine and distorted image, their approach is only useful for full-reference quality assessment.

Zhang et al. [15] presented a study of effectiveness of deep features for image quality assessment. They described the use of VGG, AlexNet and SqueezeNet for extraction of deep features and demonstrated that they are very good at prediction of image quality. They presented a new dataset of images with 'just noticeable difference' images to demonstrate the performance of deep networks. They demonstrated with VGG pre-trained model on their dataset that deep features are superior to almost all of the models utilizing handcrafted features with full-reference or reduced reference. It is to highlight that only VGG network provides a rich feature representation among the three used architectures but the more advanced architectures with deeper representation may prove more useful for the task of image quality assessment.

Gao et al. [9] has presented a deep CNN based image quality predictor. They have used VGG model pre-trained on 1 000 ImageNet categories. They extracted the deep features at each layer and trained a Support Vector Regression (SVR) on each of the deep features. These SVR are combined to form an ensemble to predict the image quality. They tested their model on several benchmark datasets and reported comparable performance. The idea behind their approach is very naïve. But the issue is that combining the deep features from all the layers results in a very large feature set size which is computationally expensive at one end and has a very large feature space at the other end. This large feature space will easily overfit the model rather than learning a more generalized form because the training database used in their experiment is relatively small.

Bianco et al. [17] proposed the use of convolutional neural networks for the task of image quality assessment. They used features extracted from the layers of CNN and also proposed their own architecture for quality prediction. Their final proposal is a deep feature extractor and these features are pooled and provided to an SVR for quality assessment. Multiple crops of an image are used and their estimated scores are averaged to provide the final quality score. They have demonstrated their model on five benchmark datasets and claimed comparable performance. Their architecture is very primitive, but they have used different learning strategies to improve the prediction performance. These learning strategies combined with a deeper and representative architecture may prove helpful in obtaining better prediction performance.

Fan et al. [18] proposed a CNN based two stage image quality assessment approach. The first phase identifies the type of distortion present in the image and the second stage contains multiple image quality assessment modules trained for each distortion type. The quality score is provided based on the distortion type identified in the previous stage. This approach can be used with success for some specific distortion types only and cannot be applied to naturally distorted images which contain number of image distortions occurring simultaneously.

Guan et al. [10] proposed a deep features based image quality assessment approach. The first step in their approach performs spatial sampling and the next

stage performs the feature extraction through CNN. There are two configurations, one performs 5×5 and then 7×7 convolution and the other performs 7×7 convolutions and their activations are concatenated to obtain the final feature vector. The next layer performs patch-wise quality assessment and weight learning for the specific patch. The last layer finds the global image quality by weighted addition of patch-wise quality. Their approach uses bilinear pooling by extracting features with different sized filters, but the depth of CNN architecture is not very deep and better representations cannot be obtained.

Bosse et al. [20] proposed a deep neural network based image quality assessment approach. The input image is divided into patches and quality is estimated for each patch and final score is obtained by averaging these quality scores. They have provided another architecture for patch-wise weighted aggregation which uses an additional fully-connected layer to learn the weight for each patch and then patch-wise weighted averaging is used for score calculation, and it has shown superior performance over the other. The patch based CNN models are easier to train and can be used on variable sized inputs by combining the scores of multiple image crops.

Bare et al. [16] proposed a specialized CNN for image quality assessment. They have used six convolutional layers along with skip connections and sum layers in their architecture. The output of last sum layer is provided to a fully connected layer of 1 024 and regression score is obtained to indicate quality for a single patch. The overall quality can be obtained by averaging over the patch estimates. The drawback of these approaches is that the proposed architecture cannot be completely trained using small image database and there is high probability of overfitting.

Hou et al. [11] proposed a deep image quality assessment approach based on qualitative scoring. Their work is based on the premise that humans prefer to provide quality judgment qualitatively rather than quantitatively, so following the qualitative approach would be more beneficial. Natural Scene Statistics (NSS) features are provided to deep belief network to learn qualitative representations which are then converted to quantitative scores for further utilization. They have presented a new direction of research in the area of image quality assessment and more work is required to improve its efficacy.

There are two approaches to perform image quality assessment,

1. handcrafted features based and
2. deep features based.

It has been demonstrated experimentally that deep features based approaches are better at quality assessment keeping in view the complexity of factors affecting the perceptual quality of an image. It can be observed from the literature that there is no consensus in the use of a pre-trained model for deep features extraction. Most of the researchers has used primeval architectures such as VGG and AlexNet, but it is not clear whether it is the best pre-trained model or some other pre-trained

model can perform better. Moreover, some authors has presented their own architectures which are inspired from AlexNet or have entirely different architecture. It is highlighted that these architectures cannot be optimally trained keeping in view the size of database they used for its training (typically containing up to 3 000 images). Therefore, we have found the need to highlight the efficacy of deep features extracted from the popular pre-trained models. We have extracted deep features at several bottleneck points and presented the three best layers for their extraction, their relative performance and the size of feature set to highlight the computational complexity. Moreover, the deep features are extracted with and without fine-tuning and important observations are highlighted to guide the reader about the architecture which provide the most quality aware features.

3 METHODOLOGY

Assessment of image quality in the absence of reference information is a complex task. No-reference quality assessment of digital images is a subjective task and there is a slight variation in quality score provided by different humans based on content and type of distortions. Therefore, subjective opinion of a number of humans is obtained and averaged to obtain MOS. Development of a machine learning model to perform quality assessment is therefore modeling of Human Visual System (HVS). The conventional approach towards this task is extraction of Natural Scene Statistics (NSS) in either spatial domain [1, 2, 3] or spectral domain [2, 4, 5, 6, 21] and then training of a regression algorithm such as SVR. The performance of the trained model is therefore based on feature set obtained through NSS. A quality aware feature set will provide better estimate of perceptual quality. HVS is a naïvely understood system and therefore its modeling through extraction of NSS is a tough task and there is always room for improvement.

Convolutional Neural Networks demonstrated expertise in the area of visual recognition in the past few years. They automatically learn the discriminatory features and perform the task of visual recognition with high accuracy and achieve/beat the human level accuracy. Some researchers [19, 17, 18, 10, 16, 11] have tried to utilize CNN for the task of image quality assessment. Their works use expertly curated deep learning architectures as well as pre-trained CNN models which are originally trained for the task of visual recognition. As the datasets with subjective score (i.e. MOS) are limited due to involvement of subjective scoring nature and the largest dataset with MOS contains 3 000 images [22]. Training from scratch with 3 000 images cannot be performed successfully on deeper models therefore shallow CNN are designed for this task which lack the impressive performance provided by CNN in visual recognition. Moreover, the pre-trained networks are originally designed for object recognition and their fine-tuning provides better performance than NSS based methods but it still needs improvement.

Recently some researchers have explored the use of activations of deep CNN layers [19, 15, 9, 17, 10] for training of a SVR to perform the quality assessment

task and achieved an impressive performance. These activations of deep layers are also referred to as deep features and have shown a good performance in several tasks which have a small dataset size and are different in nature than the pre-trained model itself. One of these researchers explored AlexNet, VGG and SqueezeNet and extracted the activations of its fully connected layers to estimate quality. Similarly, the other works principally focus on the pre-trained VGG model, as it has been demonstrated that it provides a rich representation of image content. Gao et al. [9] on the other hand extracted deep features from each layer of VGG and trained an SVR for each of them and constructed an ensemble to perform image quality estimation.

3.1 Deep Features Extraction from Pre-Trained CNNs

We have opted the approach of deep features to train an image quality assessment model. In contrast to the previous work, we have explored a number of pre-trained models to select the one with most quality aware features. We have performed an extensive experimentation on thirteen pre-trained models and extracted the activations at several deep layers of these CNN models instead of the last fully-connected layer. Table 1 provides the list of pre-trained CNN models along with three best performing layers of each model with Root Mean Squared Error (RMSE), Pearson Linear Correlation Coefficient (PLCC), Spearman Rank Order Correlation Coefficient (SROCC) and Kendall Rank Order Correlation Coefficient (KROCC). Size of feature vector for each layer is also provided to indicate the corresponding complexity of feature space.

Some observations based on Table 1 are highlighted below:

1. Final activations did not provide the most quality aware features as these are more focused on visual recognition. However, activations of the layers earlier in the CNN are better suited for the task of image quality assessment.
2. Deeper networks with more number of filters provided better image quality assessment performance as compared to networks with lesser number of filters.
3. Networks with better visual recognition performance are also better for image quality assessment task, especially the networks which have larger number of filters.

3.2 Deep Features Extraction after Fine-Tuning the Pre-Trained CNNs

Pre-trained CNNs are trained on ImageNet visual recognition dataset which has images of objects falling in 1 000 different categories. CNNs trained on these images have learned the features which are most suitable to the task of visual recognition but may not perform very well on image quality assessment task. It is therefore attempted to fine-tune these CNNs on image quality assessment datasets and then extract the activations. The pre-trained CNN architecture is modified by removing

#	Architecture	RMSE	PLCC	SROCC	KROCC	Features	Layer Name
1	AlexNet	0.7729	0.7679	0.7435	0.5512	1 000	fc8
2	AlexNet	0.7784	0.7865	0.7700	0.5760	4 096	fc7
3	AlexNet	0.7648	0.7904	0.7761	0.5802	4 096	fc6
4	Vgg16	0.8148	0.8343	0.8214	0.6309	100 352	conv5_3
5	Vgg16	0.7879	0.8433	0.8354	0.6471	100 352	conv5_2
6	Vgg16	0.7274	0.8377	0.8327	0.6478	100 352	conv5_1
7	GoogleNet	0.7966	0.7483	0.7259	0.5359	1 000	loss3-classifier
8	GoogleNet	0.6680	0.8266	0.8176	0.6232	50 176	Inception5bOutput
9	GoogleNet	0.9313	0.6877	0.8241	0.6364	163 072	Inception4eOutput
10	SqueezeNet	0.8588	0.7956	0.7835	0.5878	100 352	fire8-concat
11	SqueezeNet	0.7875	0.8139	0.8065	0.6101	75 264	fire7-concat
12	SqueezeNet	0.7655	0.8107	0.8115	0.6181	75 264	fire6-connect
13	ShuffleNet	0.9566	0.6856	0.6597	0.4724	1 000	'node_202'
14	ShuffleNet	0.9127	0.6946	0.6477	0.4717	26 656	'node_198'
15	ShuffleNet	0.8708	0.7091	0.6642	0.4820	26 656	'node_174'
16	InceptionV3	0.7198	0.8036	0.7551	0.5706	131 072	mixed9
17	InceptionV3	0.6802	0.8286	0.7921	0.6122	81 920	mixed8
18	InceptionV3	0.7190	0.8082	0.7718	0.5911	221 952	mixed6
19	DenseNet201	0.6645	0.8375	0.8127	0.6328	94 080	conv5_block32 _concat
20	DenseNet201	0.6494	0.8413	0.8165	0.6362	92 512	conv5_block31 _concat
21	DenseNet201	0.6690	0.8393	0.8115	0.6289	90 944	conv5_block30 _concat
22	MobileNetV2	0.7317	0.8050	0.7717	0.5840	7 840	block_15_add
23	MobileNetV2	0.7416	0.7897	0.7654	0.5757	62 720	Conv_1
24	MobileNetV2	0.7283	0.7923	0.7680	0.5717	15 680	block_16_project
25	ResNet50	0.6422	0.8455	0.8317	0.6458	100 352	add_15
26	ResNet50	0.6449	0.8470	0.8320	0.6455	200 704	add_14
27	ResNet50	0.6608	0.8376	0.8310	0.6405	200 704	add_12
28	ResNet101	0.6758	0.8312	0.8156	0.6273	200 704	res5b
29	ResNet101	0.6980	0.8362	0.8265	0.6416	200 704	res5a
30	ResNet101	0.6819	0.8305	0.8102	0.6274	200 704	res4b21
31	Inception-ResNet-V2	0.6851	0.8216	0.7873	0.6006	133 120	block8_9
32	Inception-ResNet-V2	0.6871	0.8199	0.7856	0.6059	133 120	block8_8
33	Inception-ResNet-V2	0.6945	0.8171	0.7776	0.5953	133 120	block8_7
34	Xception	0.7786	0.7590	0.7094	0.5251	1 000	add_12
35	Xception	0.7443	0.7855	0.7500	0.5607	262 808	add_11
36	Xception	0.7502	0.7882	0.7462	0.5616	262 808	add_10
37	NASNet	0.8858	0.7347	0.7288	0.5385	1 000	predictions
38	**NASNet**	**0.5735**	**0.8909**	**0.8982**	**0.7160**	**487 872**	**normal_concat_13**
39	NASNet	0.7095	0.8438	0.8555	0.6721	325 248	reduction_concat _reduce12

Note: Best performing layer with its corresponding scores is in bold.

Table 1. Quality assessment performance using pre-trained CNN models

the 'softmax' and 'classification' layers and adding a fully-connected layer with one neuron and a regression layer. The training is performed with a low learning rate of 0.0001. As the training dataset contains different image size than the pre-trained models, we have used random crop from the image with size complying with the pre-trained model. This will provide regularization by not letting the training architecture to learn the content of the images. Resizing is not used as the image scale is observed to affect the perception of quality. The training is performed for a total of 30 epochs with Adam optimizer. The training progress for MobileNet-v2 for 20 epochs is depicted in Figure 1.

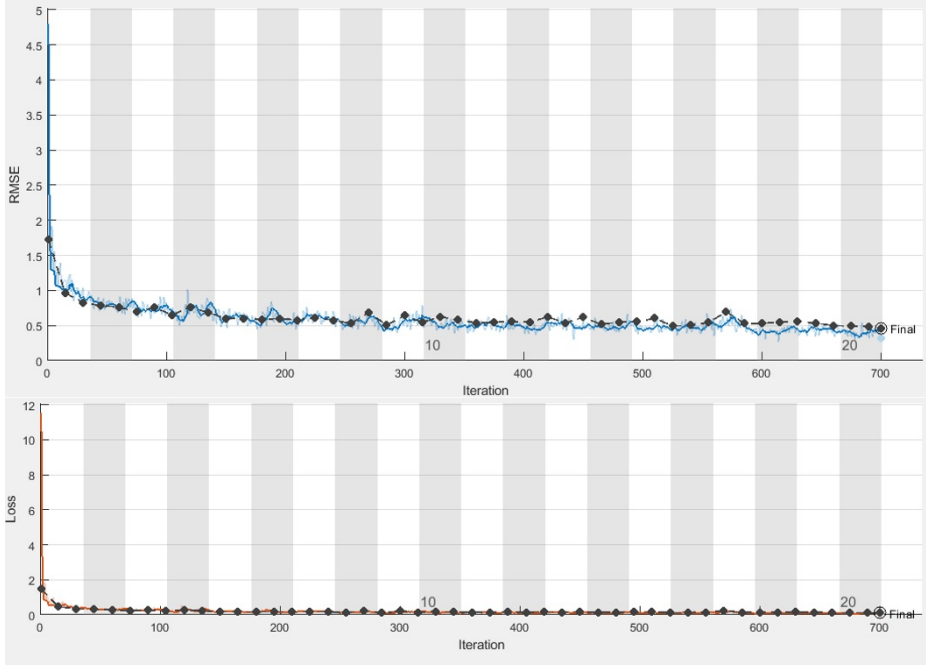


Figure 1. Training progress of MobileNet-v2

We have combined five benchmark datasets for image quality assessment to perform the fine-tuning of these pre-trained CNN models. The fine-tuning is performed with eight pre-trained architectures and the activations of different layers of trained models are obtained and trained with Support Vector Regression (SVR). The predictions from the trained model are evaluated by calculating RMSE, PLCC, SROCC and KROCC and reported in Table 2.

Observations based on results of Table 2 are highlighted below:

1. The fine-tuning of CNN models has adjusted the pre-trained models weights in a way to make it predict the image quality, so the extracted activations are quality aware features and provide better estimates.

#	Architecture	RMSE	PLCC	SROCC	KROCC	Layer Name	Features
1	VGG16	0.6407	0.8474	0.8323	0.6426	fc8	1 000
2	VGG16	0.7594	0.8121	0.7944	0.5994	fc7	4 096
3	VGG16	1.0467	0.6827	0.6672	0.4834	fc6	4 096
4	ShuffleNet	0.7391	0.8115	0.7934	0.5993	node_202	1 000
5	ShuffleNet	0.7541	0.792	0.7751	0.5805	node_198	26 656
6	ShuffleNet	0.7612	0.7912	0.758	0.568	node_186	26 656
7	DenseNet201	0.535	0.8966	0.8826	0.7022	fc1000	1 000
8	DenseNet201	0.5439	0.8943	0.8818	0.7024	conv5_block32 _concat	94 080
9	DenseNet201	0.5061	0.9085	0.896	0.7228	conv5_block30 _concat	90 944
10	MobileNet-V2	0.5784	0.8736	0.8583	0.6704	Logits	1 000
11	MobileNet-V2	0.605	0.8616	0.8438	0.6531	Conv_1	62 720
12	MobileNet-V2	0.6454	0.8425	0.8211	0.6326	block_16_project	15 680
13	ResNet50	0.5129	0.9065	0.897	0.7211	fc1000	1 000
14	ResNet50	0.5037	0.9066	0.8946	0.7167	add_16	100 352
15	ResNet50	0.4978	0.9094	0.9039	0.7307	add_15	100 352
16	Inception-ResNet-V2	0.4676	0.925	0.9187	0.7586	block8_9	133 120
17	Inception-ResNet-V2	0.4729	0.9268	0.9202	0.7596	block8_7	133 120
18	Inception-ResNet-V2	0.4684	0.9282	0.923	0.7654	block8_8	133 120
19	Xception	0.5704	0.8946	0.8833	0.6989	predictions	1 000
20	Xception	0.5712	0.8867	0.8775	0.6946	add_12	1 000
21	Xception	0.7044	0.8148	0.7958	0.6003	add_10	262 808
22	**NASNet-Large**	**0.4432**	**0.9357**	**0.9327**	**0.7768**	**predictions**	**1 000**
23	NASNet-Large	0.5117	0.9129	0.9006	0.732	normal_concat_17	487 872
24	NASNet-Large	0.5242	0.9041	0.8879	0.7139	normal_concat_16	487 872

Note: Best performing layer with its corresponding scores is in bold.

Table 2. Quality assessment performance using fine-tuned CNN models

2. More the training data, better are the deep features for quality estimation as experimented with different combinations of image quality assessment datasets and augmentation methods.
3. After fine-tuning, the last layers started providing higher performance as the model is learning the quality aware features.

3.3 Proposed Approach

Tables 1 and 2 provide the quality estimation performance of deep features using SVR with linear kernel with single image crop only. These results demon-

strate the effectiveness of deep features for the problem of image quality estimation. Some of these models have performed better or comparable to the top performing models described in literature. We have chosen pre-trained NASNet and trained it for 1 000 iterations using Adam optimizer with 0.003 learning rate. Random cropping is used during training as the NASNet input image size is different from the images in benchmark databases. Random cropping serves two purposes, firstly, it converts image size equal to the NASNet input size, and secondly, it serves as the regularization method by varying the cropping region of the image.

The feature extraction phase on the other hand provides the cropped region of the input image to generate probabilities in the last fully connected layer of 1 000 neurons which are used as features. It is to be noted that during training random image regions are used for training in each epoch and therefore performing the quality estimate at several different crops of the image will provide a better estimate. We, therefore, performed random cropping ad feature extraction for 10 random crops and an averaging ensemble is constructed to make the final prediction. The predicted image quality is therefore an average of 10 predictions obtained from different cropped regions of the image under test.

These features can be used to train a regression algorithm to provide quality estimates. It has been observed that the subjective quality scores are the mean opinion scores and therefore they tend to follow normal distribution as shown in Figure 2.

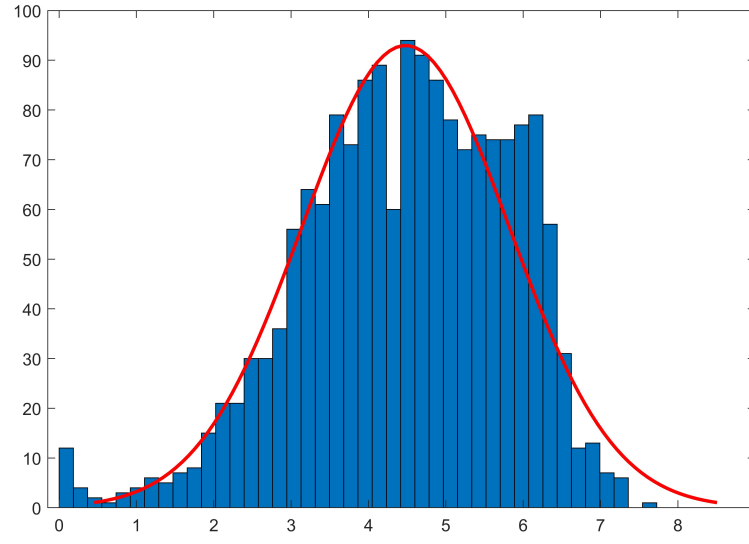


Figure 2. Histogram of mean opinion scores for TID2008 with Gaussian curve

Therefore, we decided to model the problem using Gaussian Process Regression (GPR) which is a stochastic Gaussian process-based algorithm. A Gaussian process is based on random variables and any finite set of these variables follows a joint Gaussian distribution. The standard form of Gaussian process can be defined by it mean $\mu(x)$ and covariance $C_v(x, y)$ functions and is provided in Equation (1).

$$f(x) = G(\mu(x), C_v(x, y) \tag{1}$$

where x and y are the random variables

- $\mu(x) = E[f(x)]$ and
- $C_v(x, y) = E\left[(f(x) - \mu(x))(f(y) - \mu(y))\right]$.

A number of different methods can be used to train a Gaussian process [23]. The function to make mean predictions for the GPR is provided in Equation (2) which is defined for a single test point only.

$$(\overline{\rho_x}) = \kappa_*^T \left(C_v + \sigma_n^2 I\right) y \tag{2}$$

where $\kappa_* = \kappa(x_*)$. There are different options of covariance functions for GPR, but we have used the Matérn covariance function with parameter 5/2 which is defined in Equation (3). This covariance function provided best modeling for our scenario.

$$C_v(x, y) = \sigma_m^2 \left(1 + \frac{\sqrt{5}r}{\sigma_l} + \frac{5r^2}{(3\sigma_l)^2}\right) \exp\left(\frac{-\sqrt{5}r}{\sigma_l}\right) \tag{3}$$

where, r is the distance function and σ^2 is the maximum allowed variance.

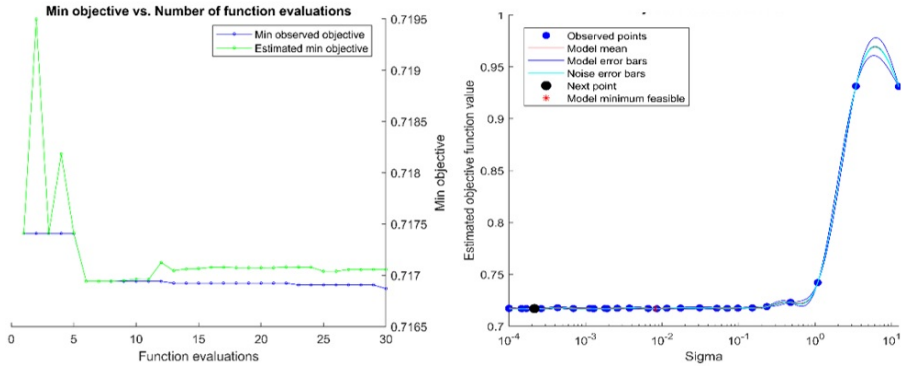


Figure 3. Left: Number of objective function evaluation vs the minimum objective function value. Right: Objective function value for corresponding value of sigma.

Optimization of hyperparameters is performed to find the optimal value of sigma, as it is crucial for the estimation of the covariance function. Bayesian optimization

is used to perform hyperparameter search and the curve for number of function evaluation is plotted against objective function value and provided in Figure 3.

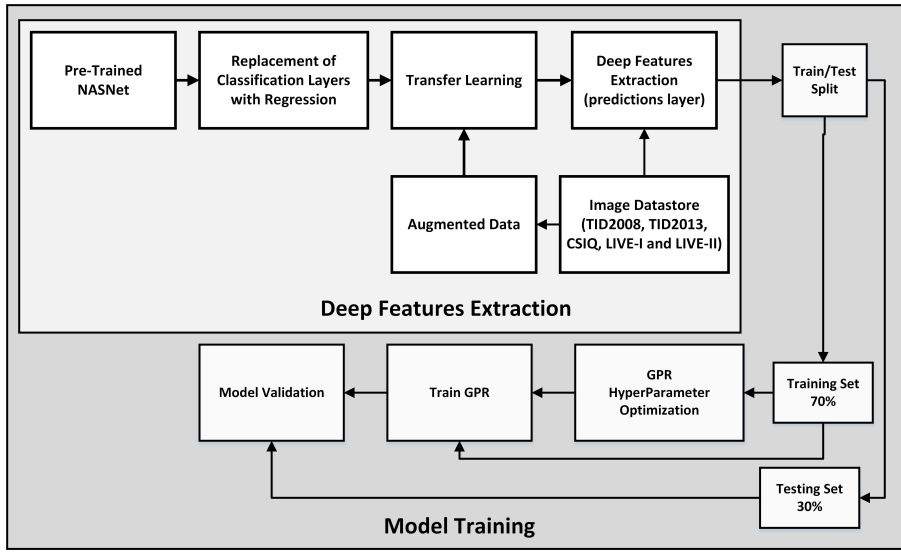


Figure 4. Model training work flow

The flow chart of feature extraction and model training is provided in Figure 4. Five datasets are used in the process of fine-tuning and model training and validation is performed for each dataset individually. Ablation study is conducted to check if the final CNN architecture used is optimal for the image quality assessment task and its results are reported in the next section.

4 RESULTS AND DISCUSSION

Some of the benchmark datasets for image quality assessment are provided in Table 3. These datasets have different number of distortion types and scoring method is either MOS or Differential MOS (DMOS). The scores are standardized using the below formula, so they fall in the same range and the cross-dataset evaluation becomes possible.

$$x_1 = \frac{x - x_2}{\sigma} \tag{4}$$

where x is the original score, x_2 is the mean of subjective scores and σ is the standard deviation of subjective scores.

The performance of the final method is measured by finding the correlation between the predicted quality score and the human subjective evaluation. Three correlation measures: Pearson Linear Correlation Coefficient (PLCC) and Spearman's Rank-Order Correlation Coefficient (SROCC) and Kendall Rank-Oder Correlation

Dataset Name	Number of Reference Images	Number of Distorted Images	Scoring Method	Range
LIVE-I	29	460	DMOS	0–100
LIVE-II	29	982	DMOS	0–100
TID2008	25	1 700	MOS	1–10
TID2013	25	3 000	MOS	1–10
CSIQ	30	900	DMOS	0–1

Table 3. Benchmark datasets for image quality assessment

Coefficient (KROCC) along with RMSE are reported for each test dataset. Table 4 provides the results of experimental testing on five benchmark datasets.

Datasets	RMSE	PLCC	SROCC	KROCC
TID2013	0.4300	0.9685	0.9717	0.8636
TID2008	0.4316	0.9487	0.9504	0.8161
CSIQ	0.0552	0.9802	0.9776	0.8696
LIVE-I	4.5840	0.9779	0.9754	0.8267
LIVE-II	3.5696	0.9752	0.9741	0.8597

Table 4. Correlation and RMSE of the proposed scheme on five benchmark datasets

The bar-chart in Figure 5 provides ground-truth values in the form of bar (green) and the predicted values in the form of stem (red) for 20 random values. Whereas Figure 6 provides the scatter plot between ground-truth and predicted values along with fitting of regression line and value of R-squared. These two plots are provided for TID2013 database and similar plots can be obtained for other benchmark database. It can be noted that the proposed model provided a good quality predicted performance and can be used as a representative model for the objective quality assessment.

4.1 Residual Analysis

The residuals are the difference between the ground-truth and predicted values and are normally plotted in the form of a bar chart. As the value of residual can be negative or positive so the bar-chart is pivoted on the x-axes with the y-axes providing the magnitude of the residuals. Figure 7 provides the bar chart of residuals for 750 (20 %) testing values of TID2013 database. The residual analysis is important in the identification of model's behavior. The residuals are checked for their normal distribution and two tests are conducted for this purpose. The histogram of residuals is plotted with Gaussian fitting in Figure 8 a) and probability plot of the residual is provided in Figure 8 b) indicating the residuals are very close to a normal distribution. The histogram is showing a symmetric distribution around zero and follows a close trend with Gaussian curve plotted for comparison. The validation of

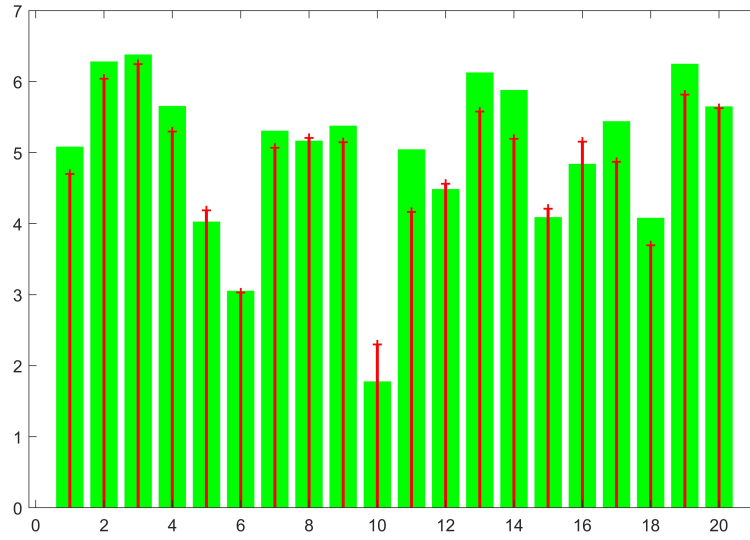


Figure 5. Model training work flow

normality test of residuals indicates that the underlying assumptions of the model are true.

4.2 Cross-Dataset Evaluation

Generalization is a major challenge in the no-reference image quality assessment. A model trained on one dataset usually performs poor on some other dataset which has a different type of distortions and uses a different experimental setup. We have therefore evaluated the performance of the proposed model by training it on one type of dataset and testing on other type of datasets. There are three categories of datasets in our experiment:

1. TID2008 and TID2013,
2. CSIQ, and
3. LIVE-I and LIVE-II.

Three experiments are conducted and reported in Tables 5, 6 and 7.

The generalizability of the proposed method can be explained due to use of a deeper architecture which provides more abstract representations of the learned features. The selected feature set is therefore the representative of image quality and provides features which are quality aware rather than content aware. Moreover, we have incorporated random cropping and other image augmentation strategies such as rotation, scaling and translation to make it robust to small variations. The scores

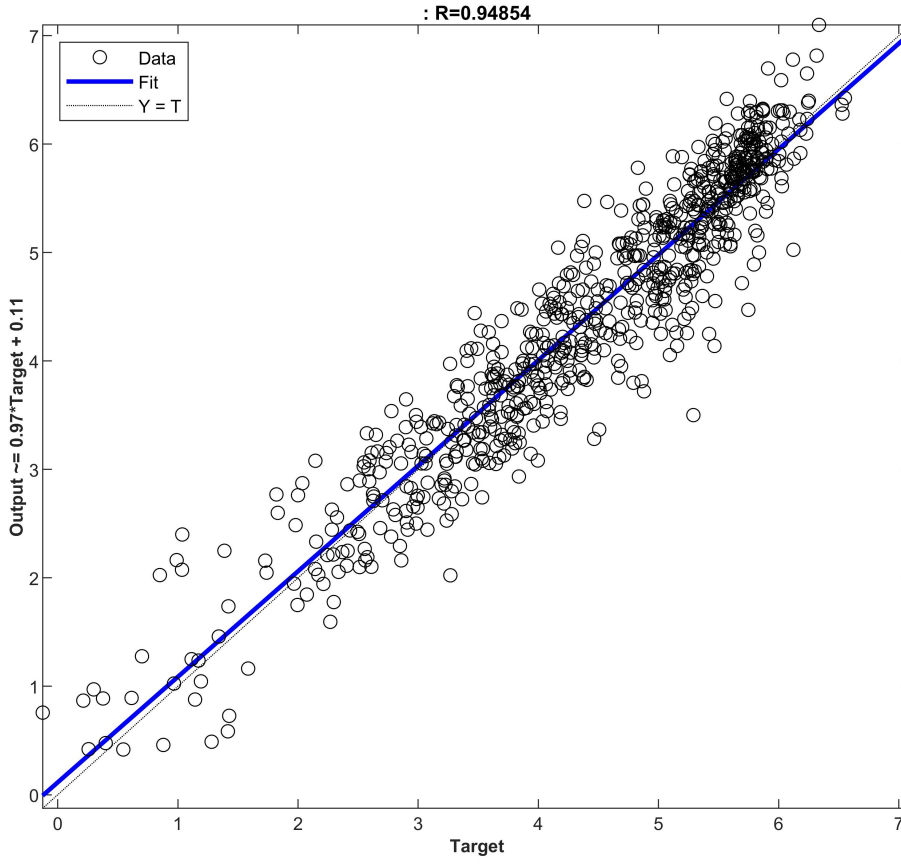


Figure 6. Model training work flow

of the different databases are standardized so the model trained on one database can predict the other database.

4.3 Comparison with Existing Methods

The performance of the proposed scheme is demonstrated in comparison to the existing methods. Ten top performing deep learning based methods are incorporated in the comparison. We have used two performance metrics PLCC and SROCC as they are the widely reported metrics and comparison is performed against three widely used datasets. The results of the comparison are reported in Table 8. It can be noted that LIVE is the most widely used dataset whereas few of the authors has reported performances for other datasets. Two best performing methods on each dataset are in bold face. It can be noted that the proposed approach has

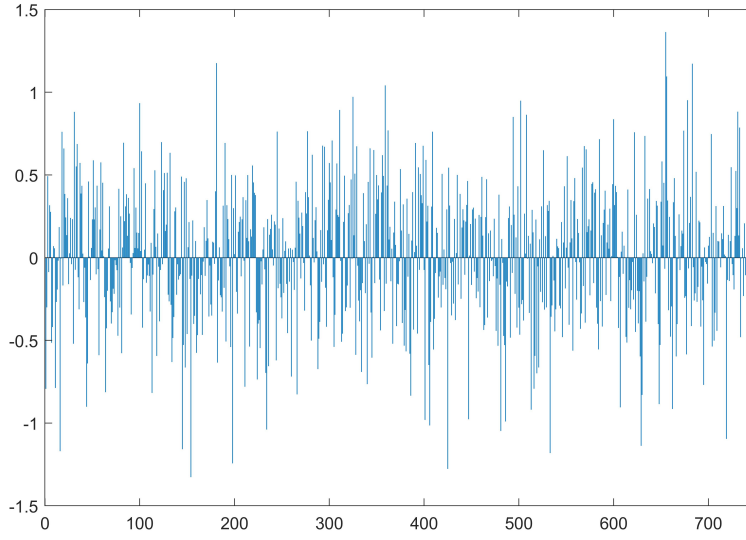


Figure 7. Bar-chart of residuals for test set of TID2013

Evaluation Measure	LIVE-I	LIVE-II	CSIQ
RMSE	5.1542	4.7514	1.2172
PLCC	0.8917	0.8815	0.8912
SROCC	0.8801	0.8798	0.8814
KROCC	0.8204	0.8102	0.8204

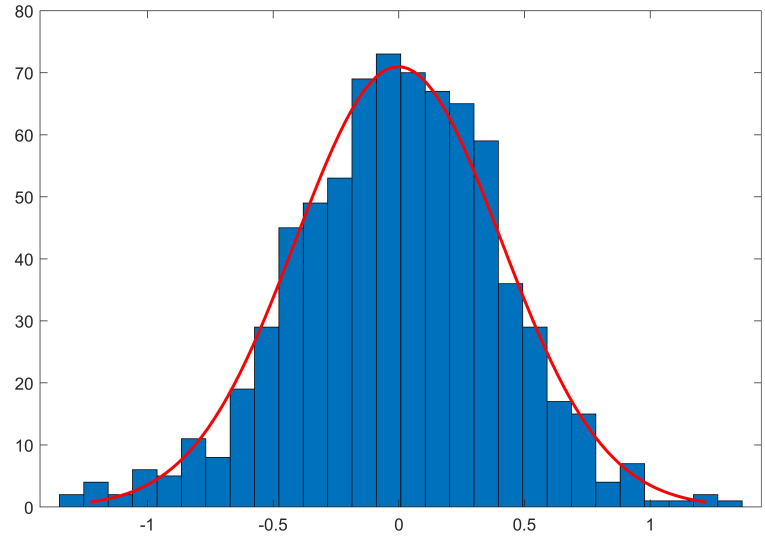
Table 5. Training on category-I dataset and testing on LIVE-I, LIVE-II and CSIQ datasets

Evaluation Measure	LIVE-I	LIVE-II	TID2008	TID2013
RMSE	7.2174	4.1572	0.7524	2.1872
PLCC	0.8617	0.8421	0.8157	0.7214
SROCC	0.8531	0.8681	0.8624	0.7189
KROCC	0.8278	0.7907	0.7124	0.5891

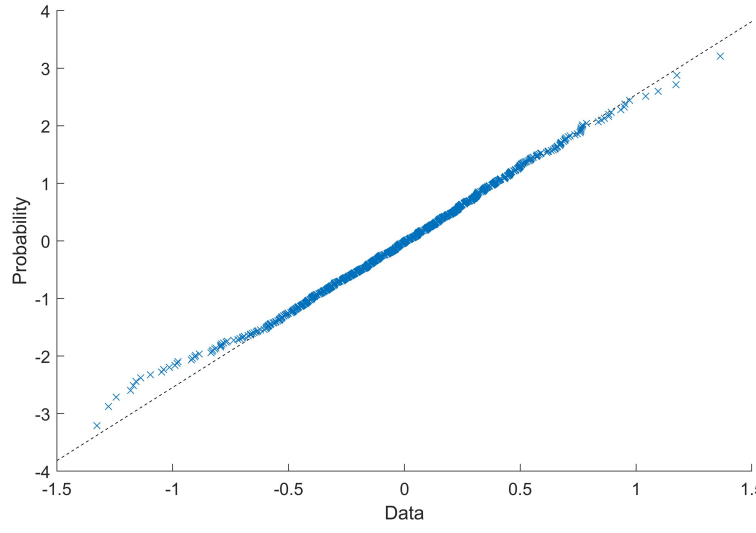
Table 6. Training on category-II dataset and testing on LIVE-I, LIVE-II, TID2008 and TID2013 datasets

Evaluation Measure	TID2008	TID2013	CSIQ
RMSE	0.7813	1.2415	2.1571
PLCC	0.8354	0.7354	0.8872
SROCC	0.8781	0.7257	0.8798
KROCC	0.7254	0.5914	0.8012

Table 7. Training on category-III datasets and testing on TID2008, TID2013 and CSIQ datasets



a) Histogram of residuals with Gaussian fitting



b) Probability plot of normal distribution

Figure 8. Normality tests using histogram of residuals and probability plots

provided the highest performance by using a single learning algorithm with multiple crops.

Dataset	LIVE	LIVE	TID2013	TID2013	CSIQ	CSIQ
Metric	PLCC	SROCC	PLCC	SROCC	PLCC	SROCC
[19]a	0.972	0.96	0.855	0.835	–	–
[19]b	0.963	0.954	0.787	0.761	–	–
[9]	0.959	0.966	0.838	0.819	0.968	0.961
[17]	0.98	0.97	0.96	0.96	0.97	0.96
[18]	0.957	0.953	–	–	0.894	0.877
[24]	0.952	0.95	–	–	–	–
[10]	0.973	0.969	–	–	–	–
[20]	0.972	0.96	–	–	–	–
[16]	0.974	0.971	–	–	–	–
[11]	0.93	0.927	–	–	–	–
[26]	0.958	0.957	0.894	0.877	0.949	0.93
[27]	0.95	0.953	0.952	0.959	0.929	0.948
Proposed	0.977	0.975	0.968	0.972	0.98	0.978

Table 8. Comparison with the existing methods

The good performance of the proposed approach can be explained by the use of a representative feature set. Perceptual quality of digital images is based on various factors such as color, contrast, noise, sharpness, artifacts and some factors which are not related to quality such as content, viewing angle and composition. Handcrafted features are therefore focused to some specific aspects of quality such as artifacts generating due to compression or some specific image processing. Moreover, these handcrafted features can model some specific classes of blur or noise but they cannot be generic to be used for all sort of impairments appearing in digital images. Deep features on the other hand are learned automatically on the basis of quality score (MOS). Therefore, deep features seem to be better candidates for image quality assessment.

Extraction of deep features which are quality aware is a tough task as the features can be quality aware only when the training algorithm is provided with a sufficient size of training data having different content. The size of training data is a very important factor when using a deep learning algorithm as these algorithms have a large number of parameters which are required to be trained, and over-fitting can easily occur if the training data is not sufficient. The limitation of the training data is slightly overcome by using augmentation which increased the effective dataset size, but a larger database will definitely be of help. The experimentation with different architectures with and without fine-tuning have highlighted the factors affecting the extraction of quality aware features, and we therefore selected NASNet-Large pre-trained model and obtain deep features after its fine tuning.

Most of the approaches highlighted in the related work used support vector machines for quality prediction which is a convenient and easy way. It provides

a reasonably good performance, but it is not an optimized algorithm in our observation. We have analyzed the MOS and observed that it nearly follows a Gaussian distribution and therefore can be modeled with Gaussian process regression. The optimization of hyperparameter for GPR resulted in final model which has high performance and good generalization. The resulting model therefore outperformed most of the existing approaches. The further improvement can be brought by training the CNN architecture with a larger and representative dataset and using the ensemble learning methods, and these two will be explored in our further work.

4.3.1 Statistical Significance Test

The Pearson and Spearman's correlation is provided in the Table 9 for comparison of the proposed scheme with the existing schemes. It is, however, worth mentioning that the absolute comparison of correlation coefficients can be sometime misleading and therefore statistical significance tests are performed to check if the propose scheme is statistically superior to the existing approaches. We have used one-sided t-test for hypothesis testing, whereas the null hypothesis is stated as the mean correlation of the row algorithm is greater than the mean correlation of the column algorithm. The hypothesis testing is performed with 95 % confidence interval. A value of '1' indicates that the row algorithm is statistically superior to the column algorithm whereas a value of '−1' indicates that the row algorithm is statistically not superior to the column algorithm. The value of '0' indicates an indistinguishable scenario of the row and column algorithm.

	[19]a	[19]b	[9]	[17]	[18]	[24]	[10]	[20]	[16]	[11]	[26]	[27]	Proposed
[19]a	0	1	1	1	1	1	1	1	1	1	1	1	1
[19]b	1	0	1	1	1	1	1	1	1	1	1	1	1
[9]	1	1	0	1	1	1	1	1	1	1	1	1	1
[17]	1	1	1	0	1	1	1	1	1	1	1	1	1
[18]	1	1	1	1	0	1	1	1	1	1	1	1	1
[24]	1	1	1	1	1	0	1	1	1	1	1	1	1
[10]	1	1	1	1	1	1	0	1	1	1	1	1	1
[20]	1	1	1	1	1	1	1	0	1	1	1	1	1
[16]	1	1	1	1	1	1	1	1	0	1	1	1	1
[11]	1	1	1	1	1	1	1	1	1	0	1	1	1
[26]	1	1	1	1	1	1	1	1	1	1	0	1	1
[27]	1	1	1	1	1	1	1	1	1	1	1	0	1
Proposed	1	1	1	1	1	1	1	1	1	1	1	1	0

Table 9. One-sided T-Test

4.4 Ablation Study

The ablation studies have been widely used in the area of neuroscience to tackle the complexities of these systems. Similarly the ablation experiments are being

used in the area of artificial neural networks owing to their increasing complexity. These experiments involve removal of a certain part of the neural network architecture to check their effect on the overall performance of the artificial neural network. These studies investigate the efficacy of the key components of the model and the experiments are done using TID2013 database. Table 1 and Table 2 provide the performance of the pre-trained CNN architectures by selecting an intermediate layer for feature extraction and discarding the layers following this layer. It was noted that without fine-tuning, the complete CNN architecture is not important for image quality assessment as the later layers have learned the features specific to object recognition task. However, the fine-tuning will make the later layers to learn the complex representations for image quality assessment and the last layer of the network performed better for image quality assessment. Table 10 highlights the performance of the selected architecture by keeping the complete architecture, removing last 41 layers and removing last 82 layers. It can be noted that the highest performance is obtained by keeping the complete architecture. Moreover, the complete architecture compacts the size of feature set, making it easy to train a regression algorithm.

#	Ablation Experiment	FeatureSet	RMSE	PLCC	SROCC	KROCC
1	Keeping complete architecture	1 000	0.4432	0.9357	0.9327	0.7768
2	By removing last 41	487 872	0.5117	0.9129	0.9006	0.732
3	By removing last 82	487 872	0.5242	0.9041	0.8879	0.7139

Table 10. Ablation experiment on NASNet-Large using TID2013 database

4.5 Computational Complexity

The experiments are performed on the Intel® Xeon® Processor E5-2687W with 512 GB SSD, 32 GB RAM and RTX 2070 GPU. The training of NASNet-Large for fine-tuning on image quality database is performed for 30 epochs for a batch size of 16 and it took almost 120 hours for training. The training of the NASNet-Large is a one-time job and feature extraction can be performed for each image in order to access the quality. The training and hyperparameter optimization of GPR took 23 minutes. The total training time is therefore 120.5 hours. Whereas in the testing phase, deep feature extraction takes 1.8 seconds per image and score prediction takes less than 120 milliseconds making it a total of less than 2 seconds per image. The testing is reported based on single core CPU only.

5 CONCLUSION

The paper presents a comprehensive insight to the use of deep features for the task of image quality assessment. As HVS is a naïvely understood subject and NSS does not perform consistently better for image quality assessment, the use of CNN

can help to overcome this limitation. Shallow CNN cannot learn the quality aware features and becomes a poor candidate, whereas the deep CNN requires a large number of training images which is not possible due to the subjective nature of image quality. Owing to the visual recognition performance of CNN, we have experimented with 13 popular pre-trained CNN models for feature extraction and eight of these were used to perform fine-tuning on a combination of five image quality assessment databases. Deep activation of NASNet-Large provided best quality estimate and a Gaussian process regression based model is trained using these features. Averaging of quality score over multiple image crops is used as the input image has a larger size then the input of NASNet architecture. The proposed methodology provided good results which are comparable with the state of the art in no-reference image quality assessment. An extensive analysis is performed to demonstrate the robustness and generalization of the proposed model.

5.1 Future Work

1. Experimental testing revealed that GPR is a good algorithm for assessment of image quality. However, ensemble learning approaches should be explored to further increase the performance.
2. The training dataset size can be improved to obtain better features, the dataset size can be increased by using weakly supervised approaches.
3. Moreover, a self-collected dataset with subjective evaluation from local users and having distortions introduced during the process of image acquisition will be used for generalization testing.
4. Combination of extracted features from a different pre-trained model may provide better performance. As the deep feature has a large size, a dimensionality reduction technique may be employed before training the regression algorithm.

REFERENCES

[1] Mittal, A.—Moorthy, A. K.—Bovik, A. C.: No-Reference Image Quality Assessment in the Spatial Domain. IEEE Transactions on Image Processing, Vol. 21, 2012, No. 12, pp. 4695–4708, doi: 10.1109/TIP.2012.2214050.

[2] Liu, L.—Liu, B.—Huang, H.—Bovik, A. C.: No-Reference Image Quality Assessment Based on Spatial and Spectral Entropies. Signal Processing: Image Communication, Vol. 29, 2014, No. 8, pp. 856–863, doi: 10.1016/j.image.2014.06.006.

[3] Sazzad, Z. M. P.—Kawayoke, Y.—Horita, Y.: No Reference Image Quality Assessment for JPEG2000 Based on Spatial Features. Signal Processing: Image Communication, Vol. 23, 2008, No. 4, pp. 257–268, doi: 10.1016/j.image.2008.03.005.

[4] Saad, M. A.—Bovik, A. C.—Charrier, C.: Blind Image Quality Assessment: A Natural Scene Statistics Approach in the DCT Domain. IEEE Transactions on Image Processing, Vol. 21, 2012, No. 8, pp. 3339–3352, doi: 10.1109/TIP.2012.2191563.

[5] Ma, L.—Li, S.—Ngan, K. N.: Reduced-Reference Image Quality Assessment in Reorganized DCT Domain. Signal Processing: Image Communication, Vol. 28, 2013, No. 8, pp. 884–902, doi: 10.1016/j.image.2012.08.001.

[6] Liu, L.—Dong, H.—Huang, H.—Bovik, A. C.: No-Reference Image Quality Assessment in Curvelet Domain. Signal Processing: Image Communication, Vol. 29, 2014, No. 4, pp. 494–505, doi: 10.1016/j.image.2014.02.004.

[7] He, L.—Tao, D.—Li, X.—Gao, X.: Sparse Representation for Blind Image Quality Assessment. 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1146–1153, doi: 10.1109/CVPR.2012.6247795.

[8] Chang, H.-W.—Yang, H.—Gan, Y.—Wang, M.-H.: Sparse Feature Fidelity for Perceptual Image Quality Assessment. IEEE Transactions on Image Processing, Vol. 22, 2013, No. 10, pp. 4007–4018, doi: 10.1109/TIP.2013.2266579.

[9] Gao, F.—Yu, J.—Zhu, S.—Huang, Q.—Tian, Q.: Blind Image Quality Prediction by Exploiting Multi-Level Deep Representations. Pattern Recognition, Vol. 81, 2018, pp. 432–442, doi: 10.1016/j.patcog.2018.04.016.

[10] Guan, J.—Yi, S.—Zeng, X.—Cham, W.-K.—Wang, X.: Visual Importance and Distortion Guided Deep Image Quality Assessment Framework. IEEE Transactions on Multimedia, Vol. 19, 2017, No. 11, pp. 2505–2520, doi: 10.1109/TMM.2017.2703148.

[11] Hou, W.—Gao, X.—Tao, D.—Li, X.: Blind Image Quality Assessment via Deep Learning. IEEE Transactions on Neural Networks and Learning Systems, Vol. 26, 2015, No. 6, pp. 1275–1286, doi: 10.1109/TNNLS.2014.2336852.

[12] Zhou, B.—Lapedriza, A.—Xiao, J.—Torralba, A.—Oliva, A.: Learning Deep Features for Scene Recognition Using Places Database. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., Weinberger, K. Q. (Eds.): Advances in Neural Information Processing Systems 27 (NIPS 2014), 2014, 9 pp.

[13] Babenko, A.—Lempitsky, V.: Aggregating Local Deep Features for Image Retrieval. Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1269–1277, doi: 10.1109/ICCV.2015.150.

[14] Zhou, B.—Khosla, A.—Lapedriza, A.—Oliva, A.—Torralba, A.: Learning Deep Features for Discriminative Localization. Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2921–2929, doi: 10.1109/CVPR.2016.319.

[15] Zhang, R.—Isola, P.—Efros, A. A.—Shechtman, E.—Wang, O.: The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 586–595, doi: 10.1109/CVPR.2018.00068.

[16] Bare, B.—Li, K.—Yan, B.: An Accurate Deep Convolutional Neural Networks Model for No-Reference Image Quality Assessment. 2017 IEEE International Conference on Multimedia and Expo (ICME), 2017, pp. 1356–1361, doi: 10.1109/ICME.2017.8019508.

[17] Bianco, S.—Celona, L.—Napoletano, P.—Schettini, R.: On the Use of Deep Learning for Blind Image Quality Assessment. Signal, Image and Video Processing, Vol. 12, 2018, No. 2, pp. 355–362, doi: 10.1007/s11760-017-1166-8.

[18] Fan, C.—Zhang, Y.—Feng, L.—Jiang, Q.: No Reference Image Quality Assessment Based on Multi-Expert Convolutional Neural Networks. IEEE Access, Vol. 6, 2018, pp. 8934–8943, doi: 10.1109/ACCESS.2018.2802498.

[19] Bosse, S.—Maniri, D.—Müller, K.-R.—Wiegand, T.—Samek, W.: Deep Neural Networks for No-Reference and Full-Reference Image Quality Assessment. IEEE Transactions on Image Processing, Vol. 27, 2018, No. 1, pp. 206–219, doi: 10.1109/TIP.2017.2760518.

[20] Bosse, S.—Maniri, D.—Wiegand, T.—Samek, W.: A Deep Neural Network for Image Quality Assessment. 2016 IEEE International Conference on Image Processing (ICIP), 2016, pp. 3773–3777, doi: 10.1109/ICIP.2016.7533065.

[21] Brandao, T.—Queluz, M. P.: No-Reference Image Quality Assessment Based on DCT Domain Statistics. Signal Processing, Vol. 88, 2008, No. 4, pp. 822–833, doi: 10.1016/j.sigpro.2007.09.017.

[22] Ponomarenko, N.—Jin, L.—Ieremiev, O.—Lukin, V.—Egiazarian, K.—Astola, J.—Vozel, B.—Chehdi, K.—Carli, M.—Battisti, F.—Kuo, C.-C. J.: Image Database TID2013: Peculiarities, Results and Perspectives. Signal Processing: Image Communication, Vol. 30, 2015, pp. 57–77, doi: 10.1016/j.image.2014.10.009.

[23] Rasmussen, C. E.—Williams, C. K. I.: Gaussian Processes for Machine Learning. The MIT Press, 2006.

[24] Liu, L.—Hua, L.—Zhao, Q.—Huang, H.—Bovik, A. C.: Blind Image Quality Assessment by Relative Gradient Statistics and Adaboosting Neural Network. Signal Processing: Image Communication, Vol. 40, 2016, pp. 1–15, doi: 10.1016/j.image.2015.10.005.

[25] Ma, K.—Liu, W.—Liu, T.—Wang, Z.—Tao, D.: dipIQ: Blind Image Quality Assessment by Learning-to-Rank Discriminable Image Pairs. IEEE Transactions on Image Processing, Vol. 26, 2017, No. 8, pp. 3951–3964, doi: 10.1109/TIP.2017.2708503.

[26] Ma, K.—Liu, W.—Liu, T.—Wang, Z.—Tao, D.: dipIQ: Blind Image Quality Assessment by Learning-to-Rank Discriminable Image Pairs. IEEE Transactions on Image Processing, Vol. 26, 2017, No. 8, pp. 3951–3964, doi: 10.1109/TIP.2017.2708503.

[27] Xu, J.—Ye, P.—Du, H.—Liu, Y.—Doermann, D.: Blind Image Quality Assessment Based on High Order Statistics Aggregation. IEEE Transactions on Image Processing, Vol. 25, 2016, No. 9, pp. 4444–4457, doi: 10.1109/TIP.2016.2585880.

Nisar Ahmed is Ph.D. student in the Department of Computer Engineering, University of Engineering and Technology, Lahore, Pakistan. His area of interest includes machine learning and digital image processing.

Hafiz Muhammad Shahzad Asif obtained his Ph.D. degree in informatics from the University of Edinburgh, UK in 2012. He is working as Chairman and Associate Professor at the Department of Computer Science, University of Engineering and Technology, Lahore, Pakistan.

Computing and Informatics, Vol. 39, 2020, 410–438, doi: 10.31577/cai_2020_3_410

MACHINE LEARNING BASED CLASSIFIER FOR SERVICE FUNCTION CHAINS

Habib Allah Khosravi, Yaghoub Farjami

Department of Computer Engineering
Faculty of Engineering, University of Qom
Alqadir Boulevard, Qom, Iran
e-mail: h.a.khosravi@stu.qom.ac.ir, farjami@qom.ac.ir

Abstract. Using service function chains, Internet Service Providers can customize the use of service functions that process the network flows belonging to their customers. Each network flow is injected into a service chain according to the flow features. Since most of the malicious applications try not to get the proper analysis by imitating some valid and famous applications, classification based on simple flow features may waste processing power by using inappropriate service chains for evasive flows. In this paper, we have explored an application-aware classification approach using machine learning methods. Using CatBoost as a machine learning method, a model is created and used for traffic classification. We have provided some statistical reports on how this approach is compared with simple flow feature-based approaches in malicious environments and how feature selection can impact classification correctness. Choosing the most suitable number of features at the right time can beat traditional approaches in classification quality and provide better results in the service function chaining environment.

Keywords: Service function chaining, classifier, machine learning, catboost

Mathematics Subject Classification 2010: 68M10

1 INTRODUCTION

The ability to classify network traffic is crucial in the operation and management of networks. Simple flow features like source and destination port numbers or layer 4 protocol are traditionally used for traffic classification, or mapping flows

into traffic classes. Since most modern applications use famous port numbers on their server-side and lots of malicious applications try to evade the proper analysis by imitating other applications features, traditional approaches have proven to be ineffective. The solution to inefficiencies of flow-based classification approaches is to harness application characteristics of the flows and making the network application-aware.

By leveraging service function chaining, Internet Service Providers are able to customize the services provided to their users, based on the type of traffic. Services are provided as predefined chains specific to special traffic types. In an application-aware service function chaining network, service chains are selected more accurately, the most related flows are steered into the chains and fewer resources are wasted. Classification of network flows in the service chaining classifier node can be based on flow characteristics or the type of traffic (Figure 1). Using the latter approach, the classifier must do a thorough analysis of the flow's packets to detect the type of traffic. Legacy traffic classification techniques are famous as the most expensive task in the network and they cannot be effective in situations where applications get frequent updates and change their signatures.

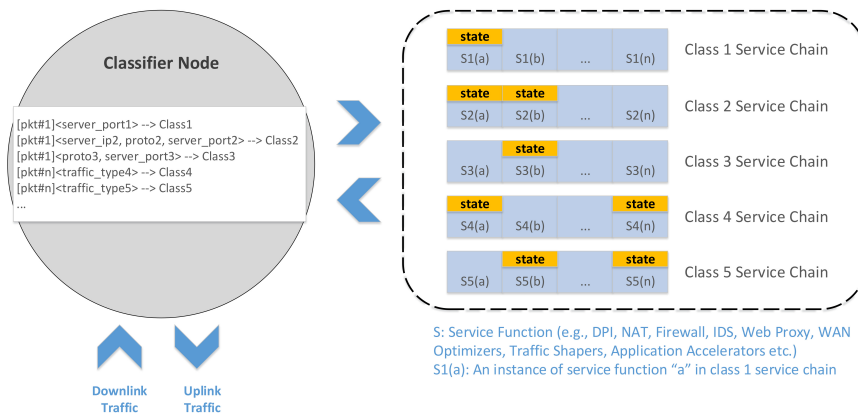


Figure 1. Service function chaining classifier node

Using machine learning algorithms in Internet traffic classification has recently received some attention [1, 2, 3, 4, 5, 6]. These approaches assume that applications send data in some patterns. These patterns can be used to classify traffic flows in different application classes. Flow features like the length of the flow, packet size, or total packet numbers can be used to find the application's behavioral pattern. Flow features collected in various observation windows, lead to results with different accuracy. Finding the best observation window is a key factor in any classification scenario [7].

This paper explores the idea of using machine learning methods in service function chaining. Using CatBoost [8], decision tree models are created to examine

classification results in different conditions. An important point to consider when using any traffic classification method in service function chaining is that the classification should be performed at the early stages of a flow's life. Delayed traffic classification can impose an unnecessary burden on service functions for processing unrelated flows. By intelligently choosing the observation window and selecting the proper feature list, machine learning-based traffic classification at the beginning of flow is possible. Based on what was said, the motivation for the current work is to find a suitable solution for using machine learning methods in service function chaining. Therefore, the contributions of the current paper are the following:

- Use of a machine learning method in a service function chaining classifier node,
- Presenting an appropriate feature list extracted from the early stages of a traffic flow's life,
- Providing solutions for the special challenges that are present when machine learning methods are used in service function chaining environments,
- Proposing a machine learning-based early classification method with a high detection rate in service function chaining.

Different observations window sizes at the early stages of a flow's life are used for creating classification models. Based on the results, the best observation window size is selected and the desired feature list is detected. The results are compared with a signature-based approach to prove their effectiveness.

The rest of this paper is organized as follows. Section 2 provides the related work. Design and implementation details and challenges are described in Section 3. The results of the experiments accompanied by a detailed explanation are provided in Section 4. Finally, Section 5 concludes the paper.

2 RELATED WORKS

This section is dedicated to the works related to the topics that are discussed in this paper. We have divided all the topics into two categories of service function chaining and application-aware classification, and machine learning algorithms and tools for traffic classification. Each category is examined individually and some of the works done in that area are introduced. We start with service function chaining and application-aware classification which is the area where the problem was originally defined. We will start with the service function chaining problem statement and also introduce some of the solutions provided to solve those issues. Then, the notion of application-aware classification is stated and after that, some of the works done in this area are discussed. When we are done with the service function chaining part, we will introduce some of the works done in internet traffic classification using machine learning algorithms.

2.1 Service Function Chaining and Application Aware Classification

What serves as a service chain is the production of a related list of network functions and, consequently, the steering of network flows among them. The use of network service functions in traditional networks is accompanied by a set of constraints. As some of these constraints are described below, each of which will be briefly described [9, 10].

The first one described here is the topological dependency. The way network services are deployed in a computer network directly affects the order in which they are used for traffic processing and creates constraints when adding or changing network services. Another constraint present in the traditional networks is the configuration complexity which is a consequence of topological dependency. Considering service function chaining in this environment, simple actions like changing the order of service chains requires changes to the topology. Consistent ordering of service functions is another constraint which is the direct consequence of the dependency on topology. Many of the network functions work in a way that they need to be deployed in a specific order. Whereas changing the order in which service functions process network traffic is complex and difficult. Another important constraint is named the traffic selection criteria which refers to the problem that all traffic on a particular network segment traverses all service functions whether or not the traffic requires those services. Besides the aforementioned constraints, there are some others like constrained high availability, application of service policy, etc. that make the use of service function chaining inevitable. Service function chaining provides a set of solutions to resolve the problems existing in traditional networks and to make the network management more elastic and robust [10, 11].

The three solutions provided by the SFC working group at IETF to overcome the above constraints are service overlay, service classification, and SFC encapsulation. Service overlay is about creating service functions connectivity built on top of the existing network topology. It will be possible to create an arbitrary topology for connecting service functions in a required topology. Service classification and reclassification procedures are used to select which traffic enters an overlay and alter the sequence of service functions applied to traffic. Finally, using SFC encapsulation enables the exchange of metadata in the data plane between different components of the service chain [10, 12, 13, 15].

One of the challenges about traffic classification in service function chaining is using application data to select the service overlay where a network flow is going to be processed. In other words, instead of using the mere port numbers or the flow's network-level characteristics for traffic classification, information about applications involved in the generation of the traffic is used. Therefore, service overlays are selected more intelligently, and using service chaining's maximum potential becomes possible [14]. A deep packet inspection (DPI) function can do the task of application-aware service classification in a service function chaining scenario.

One of the works that has made use of the output of the traffic detection function (DPI service function) in creating service function chains is the StEERING [16]. The result of processing the traffic in DPI and the detected applications might be used in service chain selection for a network flow or it may alter the chain already selected for the flow. As the detection of applications in a DPI function might not be possible with the first received packet in flow, passing the packets to a new chain after the detection is finished might result in an incomplete analysis in some service functions.

SIMPLE is another solution that provides the definition of traffic routing policies between network services and translates these policies into traffic forwarding rules [17]. Different traffic classes along with their corresponding service function chains are defined by the network admin which are used for creating traffic forwarding rules based on the current state of the underlying network. Resource manager, dynamics handler, and rule generator are the three key components in SIMPLE system. Resource manager is in charge of providing the state of resources available in the network for the rule generator component which is the place where traffic forwarding rules are created. Dynamics handler detects the changes in traffic which are made by the service functions and provides the necessary information for the rule generator to create new forwarding rules. This component uses packet payload to find the connection between two distinct forms of one flow in the network.

There is also another solution for using the result of traffic detection in software-defined networks that has utilized the idea of delayed traffic flows [18]. When a new traffic flow is received at the ingress to the network, the packets are delayed and they are mirrored to a DPI box for the traffic detection process. When the result is provided, the traffic scheduling module installs forwarding rules in the network and the flow packets are allowed to pass. Clearly, the solution of delayed network flows cannot be used in high-performance networks where a huge amount of network traffic is passing. A solution that has incorporated the traffic detection procedures in the network controller is also available, it cannot respond to scalability requirements in high-performance networks [19]. Other work that has mentioned the use of traffic detection tool results to reroute the traffic flows in the network has also ignored the importance of all packets being processed in the network functions [20, 21].

Using DPI as a service is another work done in this area. The idea is to remove the deep packet inspection function from service functions like IDS or Firewall and deploy it in a separate service function. Other services in a service chain can leverage the metadata provided by DPI service function without running deep packet inspection procedures themselves. Actually, what this work states is that deep packet inspection is done in multiple service functions in a single chain and it is the main reason for processing delay in each service. By using the idea proposed in this work, deep packet inspection procedures are only executed once in every service chain where other services use the results of deep packet inspection analysis in that service [22].

2.2 Machine Learning Algorithms and Tools for Traffic Classification

Machine learning refers to a set of techniques and algorithms that are able to learn from data and make predictions on data. Such algorithms overcome the process of following strict rules and instructions provided in a program by making data-driven predictions and decisions, through building a model from training inputs. Machine learning techniques can be split into three categories of unsupervised learning, supervised learning, and semi-supervised learning based on the amount of labeled data used in their training process. Since only the use of supervised learning techniques is present in the current work, we will only focus on the works which have used supervised learning techniques for traffic classification.

A supervised machine learning algorithm uses a labeled dataset for training. Every instance in the dataset contains the same set of features which is mapped to a given known label [23]. The result of the training phase is a model which can be used to predict the label for unknown set of data with the same feature list. To the best of our knowledge, there are no works that have specifically leveraged a supervised machine learning algorithm in a service function chaining classifier node. Because of that, we only discuss some works that have used a supervised machine learning for the general idea of traffic classification.

One of the works that rely on behavioral features of internet applications which are present at the start of the flow and do not use the packet payload to extract any features, claims to have achieved 99.8 percent of accuracy. Considering a set of 248 flow features from the beginning of individual network flows with different observation windows size, it leverages feature selection algorithms to find the best subset of features. Finally, the feature subset is used to train a model and the model is used for classifying unknown flows. The authors believe that if the traffic classification system is going to work near real-time with a considerable throughput, an appropriate and small set of features must be selected from a small number of packets in a limited duration. This work has used C4.5 decision tree since it had a low complexity and finally, they believe using features extracted from 5 or 6 packets in the network can achieve the highest accuracy [24]. The complete 248 features are detailed in [25].

A solution comprised of signature-based and machine learning methods is also proposed that claims 99.7 percent of accuracy in traffic classification [26]. After labeling the dataset using Snort and using multiple techniques for extracting the best set of traffic features, the model is created using a multi-classifier and it is used for traffic classification. Although the authors have mentioned that they have not placed a constraint on feature extraction window size, it seems like the final feature list is dependent on the full trace of a network flow.

There is another work that has considered the use of statistical features of a flow for traffic classification using machine learning. This work combines unsupervised and supervised machine learning algorithms to provide a method for internet traffic classification. It extracts traffic features and clusters them using an unsupervised machine learning algorithm. The result is used as the training data in a supervised

machine learning algorithm and the output model is finally used for classifying unseen traffic flows. They claim that the method has an accuracy of 90 percent for classifying unseen traffic flows. A set of five and eight features is selected for unidirectional and bidirectional flows, respectively. Both feature lists can be provided at the end of a flow [27].

Another solution that has proposed a two-phased machine learning approach by using an unsupervised machine learning method for feature extraction and using a supervised machine learning algorithm for traffic classification has provided a feature list completely dependent on the full flow trace [28].

Taking into account the related works in the two sections related to service function chaining and machine learning techniques, the need for doing a similar classification work in service function chaining environments is felt. This work is allocated to the idea of using machine learning tools in the classifier node in service function chaining.

3 DESIGN AND IMPLEMENTATION

In this section, the proposed approach for traffic classification in service chained environments is discussed. Starting from the nature of malicious traffic and inability of the traditional approaches to classify this type of traffic, continuing with presenting the architecture where the proposed approach is used and finishing with the implementation details.

3.1 Design Challenges and Requirements

Traditionally, applications used proprietary port numbers on their server-side. Network traffic was mostly not encrypted and protocols like HTTPS were not so commonly used. For example, port number 80 was reserved for HTTP protocol and the protocol's traffic was plain text. Nowadays, traffic is mostly encrypted whether the application is malicious or not and port hopping is a common technique to evade network analysis devices like firewalls or IDS/IPS. Malicious applications are increasing every day and can change their traffic pattern easily.

In this situation, using simple port numbers or pattern matching approaches for detecting applications is inappropriate and the decision made on such knowledge cannot be trusted. So, deep packet inspection devices that utilize signature-based approaches that need heavy processing are not suitable for some scenarios. Besides, the nature of applications' traffic is always changing and signatures used in the aforementioned approaches need to be updated constantly and cannot be sufficient in all cases. This makes it challenging for service function chaining scenarios to use application characteristics in traffic classification. Therefore, malicious network flows may find the chance to stay out of sight of appropriate analyzing services by imitating benign applications' characteristics. Besides, service resources are wasted analyzing the wrong imitating flows. By mentioning signature-based or pattern-based schemes, this paper refers to the approaches that utilize a set of predefined

rules and signatures for traffic classification. Signatures are created by following the strict rules and instructions provided in an application, after analyzing the application's traffic. The classification of a network flow is performed by comparing the details of multiple packets in the flow with the predefined set of rules and signatures. Examples of the commonly used signatures are the server-side port numbers used by Internet applications and string patterns present in the applications' data.

Recent research studies have shown that machine learning approaches can be leveraged in the process of network traffic classification. Machine learning can be used to construct models that learn to decide whether a network flow is malicious or not, directly from the data and without any predefined rules [29]. Each Internet application's network traffic has some characteristics and behavioral patterns that can be used to predict other unknown network flows from the same application. By extracting historical or application-specific features from labeled network flows and feeding them into a supervised learning algorithm, models can be created to classify network flows.

Results show that some approaches have achieved better than 95 percent of accuracy by using machine learning algorithms [2, 24, 26, 28]. In this work, we have used some historical and application-specific features of network flows and created a model to be used in a classifier node in service function chaining scenarios. By mentioning machine learning-based schemes, the paper refers to the category of approaches that use supervised machine learning methods for training a classifier based on an application's available network traces. Instead of comparing details of a flow with predefined signatures, multiple flow characteristics are extracted and used for making a data-driven prediction about the flow.

One of the main goals in applying service function chaining in a network is to differentiate the services provided for network flows. As most machine learning-based approaches proposed for traffic classification are based on features extracted from a full flow, using machine learning methods in a service function chaining classifier is challenging. Delayed traffic classification can cause unnecessary burdens on unrelated services and changing a flow's path in the middle of its life results in an incomplete state in some service functions. Therefore, early traffic classification is a necessity in a network applying service function chaining.

3.2 Proposed Solution

There are a lot of features that can be extracted from a total flow. Lots of them need the flow to be finished to become available [25]. But this cannot be achieved in service function chaining scenarios where real-time classification is needed. In this case, flows must be classified as soon as possible to be analyzed in the corresponding service chains. Otherwise, since some stateful services may need to inspect all the packets in a flow, the classifier has to enable multiple services that belong to different classes for packet inspection. Technically, some services need to store state for each flow processed by them. When a new flow enters a service chaining network while it has not been mapped into a class, it needs to be injected into a default chain of

multiple services belonging to different classes. This way, the state stored in those services is complete even after classification. The natural result of using a default service chain is that the resources on some services are wasted on unrelated packets. Figure 2 illustrates two stages of a flow's life in the network, before and after the classification. The packets in the flow are steered into the default service chain until the successful classification results are prepared for the flow. After that, only the services corresponding to the flow specific service chain are present.

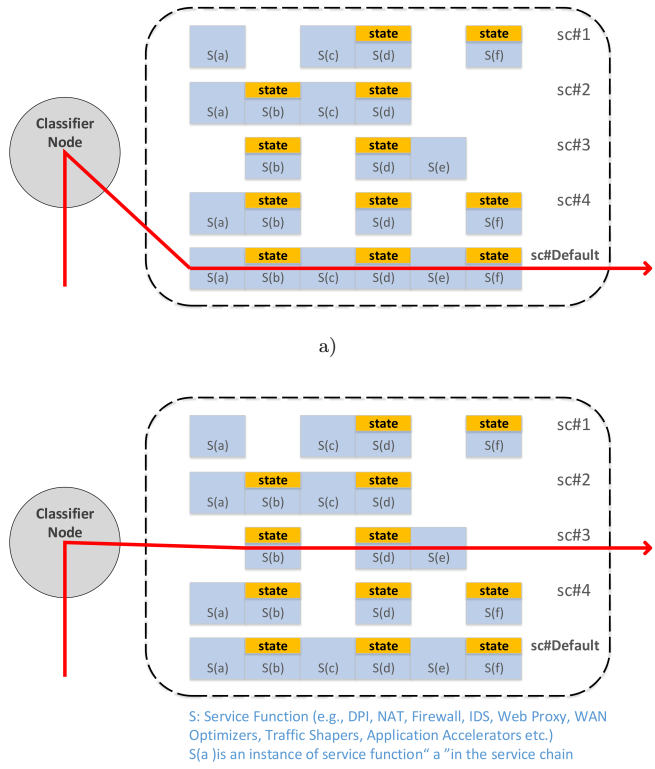


Figure 2. Illustration of a flow's life in a network a) before and b) after flow classification

Some previous work has mentioned that considering a total of 5 or 6 packets from the start of flow can achieve the highest possible accuracy for traffic classification [24]. In service function chaining scenarios there is a tradeoff between the number of packets used for feature extraction and the load on the services belonging to each chain (as depicted in Figure 3). So, finding the best point for flow classification (feature extraction windows size) is the key to using machine learning algorithms in such environments.

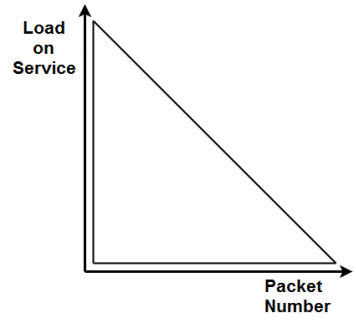


Figure 3. The tradeoff between the packet number and the load on services

In this work, the list of features used for traffic classification is categorized into historical and application-specific features. The list of historical features contains the 5-tuple characteristics of each flow which are the source and destination IP address, transport layer protocol, source, and destination port numbers. By historical it refers to the fact that network flows from specific client locations to specific server locations may happen again. It is true most of the time that a triad of server IP address, protocol, and the port number corresponds to a specific application server on the Internet. If we consider the habits of the clients in using the internet, the 5-tuple feature list may lead to a good prediction result for unknown flows.

In the application-specific feature list, besides the list of historical features, the features related to the application traffic patterns have been considered. Because of this, only the packets containing at least one byte of data have been used for feature extraction. By configuration, features from up to three data packets are used for traffic classification. The list of features extracted from each data packet is as follows:

- the size of transport layer payload (application data),
- the size of the captured data from network,
- the packet direction (from client or reverse),
- packet number in flow,
- TCP header push flag,
- packet entropy.

The size of the application data in the first packets may contain important information about the application. For example, this data is normally small for the first packet in an SSL flow. Besides, some malicious applications may try to segment data packets to a specific size or create messages of a specific size that can be considered as a pattern. To consider the size of other network layers' headers, the size of the captured data is also considered. In most client/server applications, the client sends the first data packet but this may be different in some other applications.

Also, it may be done by some malicious applications. This is the reason that data packet direction is considered. To consider the number of packets without data, the number of packets in the flow is also considered. TCP header push flag is also set in some applications' data packets which is also considered. Finally, packet entropy is also used as a feature that can specify whether the data is encrypted or not. This is an important feature for detecting some applications.

3.3 Technological and Implementation Details

The purpose of the experiments is to show how machine learning can help to classify network flows in a malicious environment where a good percent of the flows try to evade detection or analysis by imitating some famous applications' patterns. Also, to specify the best way to use machine learning algorithms in a service chaining environment. For this reason, nine application classes of Unknown, HTTP, DNS, HTTPS, FTP, Telnet, SSH, SMTP, and QUIC are considered. There are lots of malicious applications that try to evade detection by leveraging some of these famous protocols' characteristics (using their default port number for example). The classes have been chosen to somehow simulate the diverse nature of the Internet. All the experiments done by using machine learning algorithms are also compared with experiments done using a signature-based scheme where only port numbers are used for the classification of flows. In the end, some other experiments using pattern matching techniques are also presented.

CatBoost is a machine learning method based on gradient boosting over decision trees. Gradient boosting is a machine learning technique that can be used in classification problems by creating an ensemble of classifiers, typically decision trees. Gradient boosting trees have some properties that can perform excellently in network traffic classification [30]. In this work we have used CatBoost to create the model used for traffic classification. CatBoost performs well in the classification of traffic into several classes and it can be a great choice to be used in this work [31].

30 Gigabytes of network traffic has been captured from a company's network where a vast amount of malicious applications are active. The company provides Internet access for a group of people comprising scientists, students, and other people. The data was captured with a rate of 21 Mbps for more than four hours at peak time. After cleaning the data by removing TCP flows without SYN packet and removing flows with no data packets and also, removing duplicate flows, about 450 thousand flows are left. These flows are fed into two available DPI modules (one of them is a commercial DPI and the other one is open source nDPI [32]) and they are all labeled with classes mentioned above. Traffic flows in this data set belong to all of the classes with different proportions. This labeled data set is used as the train data to be fed into CatBoost and to create the model.

Two smaller traffic captures have also been prepared for running the experiments. One of them is captured in a network where a mix of multiple traffic classes is active and some malicious flows exist in it. The other one is captured in the same network where the training data was captured. They are named mixed traffic and

malicious traffic, respectively. By malicious, it refers to the fact that the rate of using deceptive techniques by applications present in this network is higher.

For each traffic capture the following set of experiments is performed. At first, the traffic is fed into a signature-based classifier where only port numbers are used to detect flows (the port number is used as a representative for signature-based methods). For each traffic class, its default port number(s) is used as the signature. After the classification of a flow, packets are sent into a service chain specific to that class where they are processed. Besides the signature-based classification, multiple tests are done using machine learning algorithms. One test is done only using historical features. After that, three tests are done with application-specific features extracted from one, two, or three data packets.

There is a difference between a flow classified in Unknown class and an unclassified flow. For a flow that is classified as unknown, we can say for sure that it does not belong to any other class. But, for an unclassified flow, nothing can be said about its class. Therefore, for flows classified to Unknown class no services are used and the traffic is passed unprocessed. The results of the tests are compared to two main parameters. The first one is the quality of classification of flows and the second one is the load on each service chain and consequently, on services.

4 EVALUATION RESULTS

In this section, we investigate the results of the tests mentioned above for the three metrics of accuracy, recall, and precision. Accuracy is about the number of flows predicted correctly in each test. Technically, it specifies the number of flows correctly predicted in a class and correctly predicted not in a class. Recall considers the number of flows in each class which is predicted correctly. In other words, it specifies the number of flows that are actually in a class and have been correctly predicted. Finally, precision specifies what percent of the positive predictions are correct. Figure 4 presents the formulas for calculating accuracy, recall and precision. As mentioned above, two data sets are prepared to run the tests. Before we run the tests, each data set is examined using the same DPI modules that we used for labeling the training data and these results are used for evaluating each metric. After that, each data set is once tested in a signature-based scenario and then in multiple machine learning scenarios. The average value of accuracy, recall, and precision are calculated for each type of test and when the result of all tests is examined, the average values are compared to find the best approach for traffic classification. In the end, the amount of traffic load on service chains is also investigated.

Figure 5 presents the results of the signature-based tests on two mixed and malicious data sets. Mixed data set refers to the traffic captured in an environment where a mix of benign and malicious applications are active with a broad list of applications and the malicious data set refers to the traffic captured in an environment where a vast amount of active applications are malicious. The results for mixed and

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

Figure 4. Formulas for calculating accuracy, recall, and precision

malicious traffics are distinguished using their colors. Results of accuracy, recall, and precision are all depicted in Figure 5.

As can be seen in Figure 5, all application classes except HTTPS and Unknown result a high accuracy of more than 90 percent. This means that with respect to those classes, most traffic flows are predicted correctly in a signature-based approach. The reason for a lower accuracy in HTTPS class is the fact that this port number is the most popular port number between malicious applications and all the applications that need to hide themselves from the sight of network analysis devices. The accuracy for the Unknown class is also low because the failure in classifying HTTPS traffic results in decreased accuracy for the Unknown class. A large number of Unknown flows must be classified in HTTPS class which is the result of evasive techniques employed in malicious applications.

Like the accuracy, we can see a high value of recall for most classes except the Unknown class. A low value of recall for class Unknown implies a high percentage of unknown flows which are incorrectly classified in other classes. The value presents the percentage of correctly predicted Unknown flows. Aside from the Unknown class, we see a lower recall for class HTTP compared with other classes. This implies that also for HTTP traffic there are some flows that have not used their default port number which is port number 80. This must be true since there are many HTTP servers running on port numbers 8080 or some other ports similar to this one. For HTTPS class we see a very high value of recall. The reason for this one is that almost all HTTPS traffic has used its default port number and nearly all of these flows are predicted correctly. Since some of the traffic classes were not present in the malicious traffic data, they are depicted without the value of recall.

By looking at the precision results in Figure 5, we will come up with more interesting conclusions. A low value of precision for a specific class is the result of some other flows incorrectly predicted in this class. As we can see, the class Unknown has a high value of precision while classes like HTTPS, SSH, or QUIC have resulted in lower values of precision. This means that a large number of Unknown flows have incorrectly been predicted as other classes. This is where we can see the track of malicious applications trying to mitigate other application properties. Classes

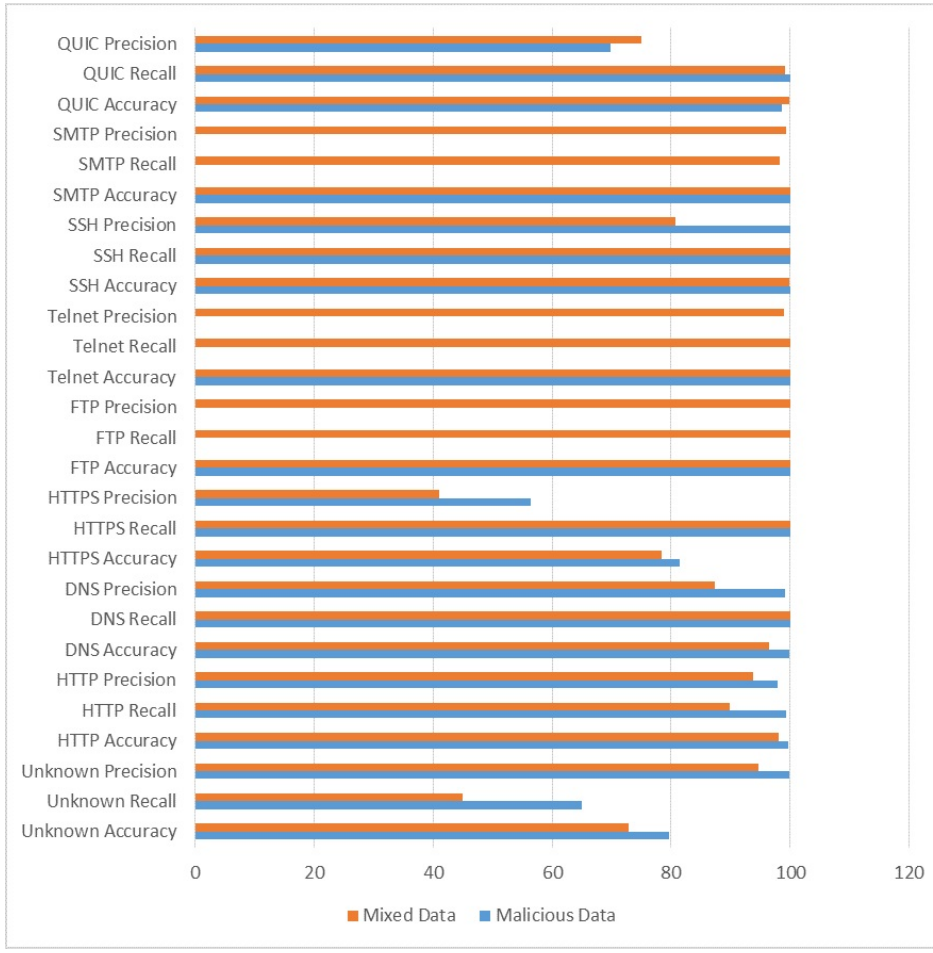


Figure 5. Results for signature-based scenario

without a precision value are also present since no traffic flow has used their default port number.

Besides the above discussion, we can see lower values in almost all metrics for the mixed data compared to the malicious data. This can emphasize the fact that the same trend of using evasive techniques is active in both networks.

After discussing the signature-based detection scheme, it is time to consider the results of machine learning-based experiments. Machine learning-based detection using a historical feature list (flow characteristics as mentioned before) is considered first. When speaking about a historical feature list, tracking the type of flows from specific clients to specific servers on the Internet is considered. When the model is created based on the type of flows originated from some clients to servers on

the Internet, it can predict unseen flows from the same clients to the same servers based on what it has seen historically. Based on the similarities of the new flows' characteristics with the flows used in model creation, new flows can be classified.

As mentioned in the previous section, the historical feature list is the 5-tuple of source/destination IP address, protocol, and source/destination port numbers extracted for each flow. Figure 6 compares the results for the machine learning-based classification using historical features in both mixed data and malicious data.

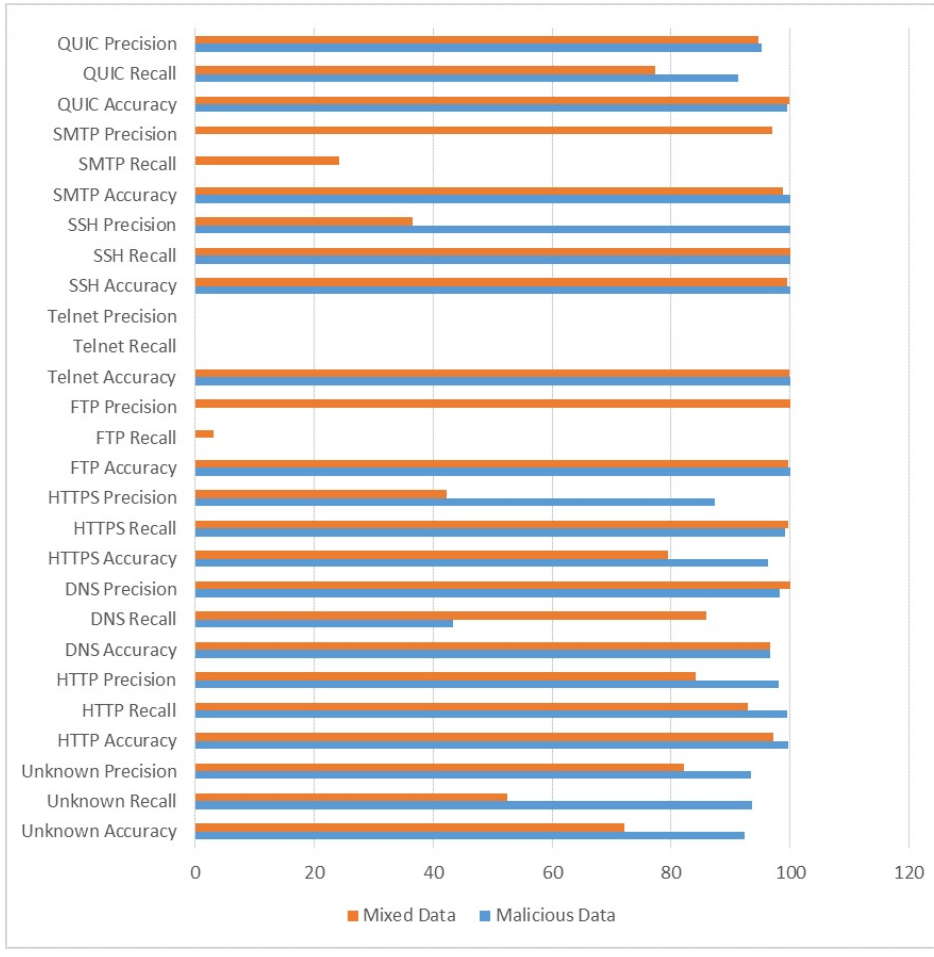


Figure 6. Results for historical features based scenario

Before we speak about the results presented in Figure 6, it should be reminded that the data set used to create the model for all the experiments was provided in a network with malicious traffic where lots of flows on famous port numbers are of

other unknown protocols. So, we can look for the effects of such a phenomenon in all experiments.

As it can be seen in Figure 6, the accuracy results for all traffic classes in malicious data are more than 90 percent while we see lower values of accuracy for some classes in mixed data. Since this experiment only considers the historical data for creating the model and the malicious data set was captured in the same networks as the training data, the higher accuracy for malicious data is justifiable. A high value of accuracy for the malicious traffic implies that the use of historical information in a network may be effective in application classification.

We can see a similar result comparing recall between mixed and malicious traffics. For most of the classes the value of recall for mixed traffic is less than the value of recall for malicious traffic. We believe this must also be the result of the difference between networks where training data and the mixed data are captured. As an exception, we can see that the DNS class has resulted a better recall for the mixed traffic data. Clearly, the list of DNS servers in all the networks all over the world is similar. This is the reason the amount of recall is not lower for mixed data. Also, the presence of a lot of malicious applications in the malicious network may cause a lower recall for that case since there may be some traffic flows trying to mitigate DNS traffic while they are not actually DNS.

Following the same pattern, we see lower precision for the test with the mixed data. Lower precision shows a lower percentage of correct positive predictions which can be justified considering the different networks mixed and malicious traffic where captured.

Considering the mixed data traffic, we can see that the precision for HTTPS and SSH classes is lower than the other classes. Comparing the result of precision in these classes in the mixed data with the same classes in the malicious traffic, we see much better results for the malicious data. This emphasizes the fact that the characteristics of these application classes are desirable with malicious applications and the lack of historical information about the traffic can result in lower precision when predicting application classes.

We saw the effects of using historical features in flow classification both for mixed and malicious traffic. Now we want to enter some features related to the actual application payload into the experiments. These features are listed in the previous section and they are categorized based on the number of data packets where the flow classification takes place. First, we look at the results where only the first data packet is considered. Figure 7 presents the results for this experiment.

It can be seen that adding some application payload features to the model can do the magic where the accuracy results for all classes in both data sets are more than 90 percent. This result presents the power of a good machine learning scheme when application-specific features are considered for classification. We can see that when application payload characteristics are added to the model, the difference between the accuracy for mixed data and malicious data is reduced.

Now that we have seen the result of using application-specific features in flow classification, it is time to increase the number of features by using more data

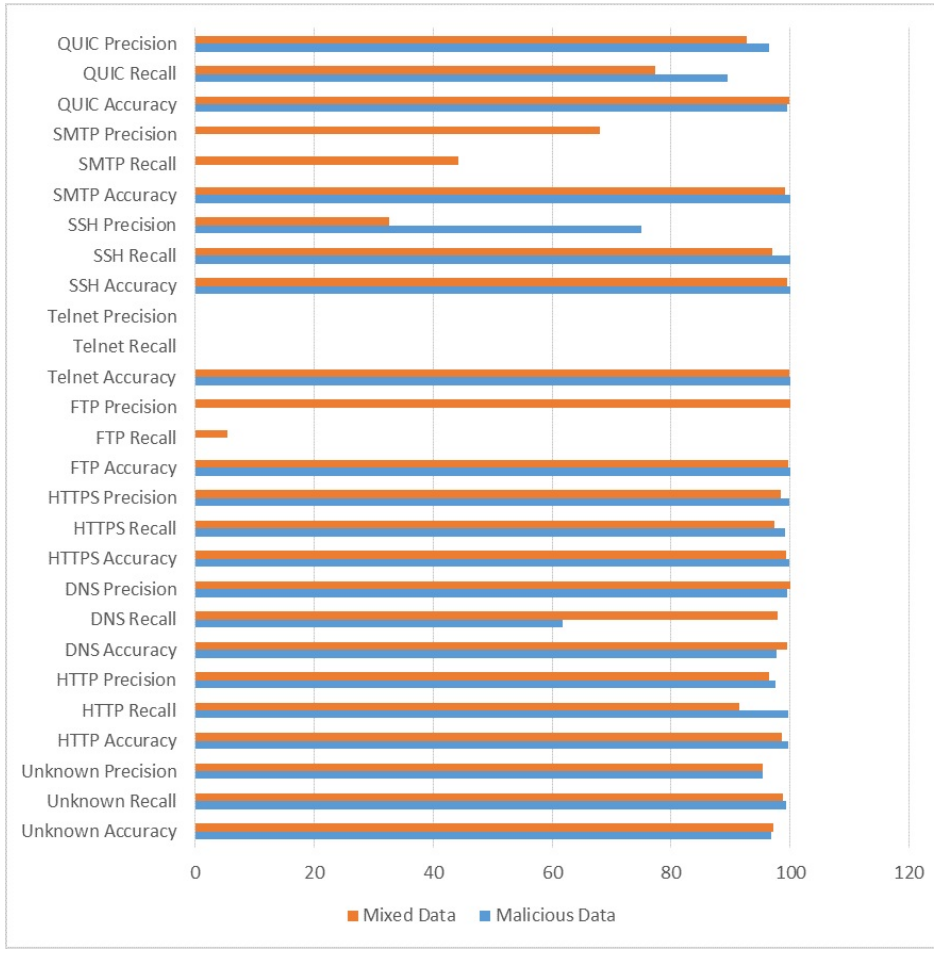


Figure 7. Results for application-specific features based scenario with one packet

packets in flow classification. Figure 8 presents the classification results when the classification is done as the second data packet is received in the flow and features related to the application payload are from the first and the second data packets.

Before speaking about the results presented in Figure 8, it should be noted that in any traffic data set, there may be some flows with only one data packet. So, when considering the second data packet for flow classification, some of the flows remain unclassified until they finish. The results presented in the above table are only considering the classified flows and unclassified ones are ignored. So, if for example it says that 10 percent of the flows are in HTTP class, it is referring to the set of classified flows and not the unclassified ones.

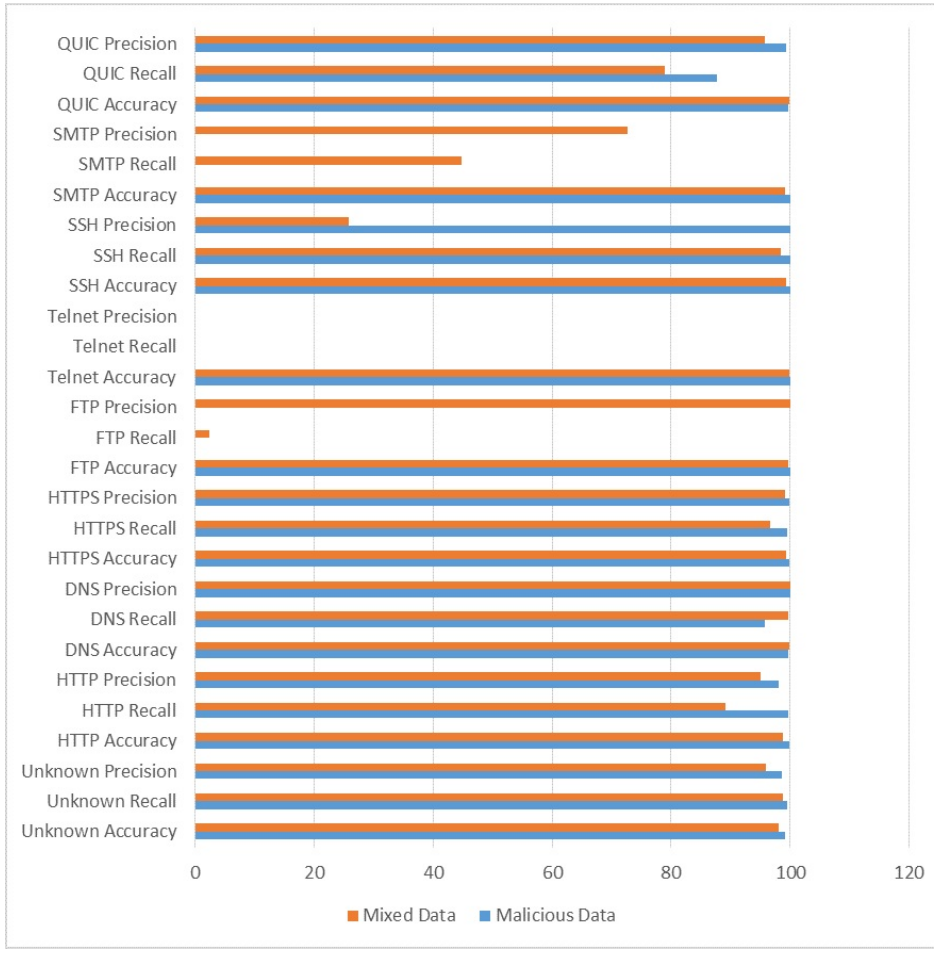


Figure 8. Results for application-specific features based scenario with two packets

Considering the results presented in Figure 8, we see the accuracy of more than 95 percent in all application classes. Compared to the results of experimenting with only one data packet, we can also see an increase in the value of recall for almost all classes. This result can also be observed for precision values. Some incompatible results for some of the classes (like the reduced recall for SSH) may be judged by the one packet flows which are omitted from the results in the current figure.

Finally, we want to consider one more data packet in the flow classification procedure. As the third data packet is added to the procedure, more features specific to that packet are added to the model. Besides, as mentioned above, the natural consequence of waiting for more packet until traffic classification, is the reduction in

the number of classified flows. Figure 9 presents the result of the experiment using three data packets where only the classified flows are considered to calculate the values.

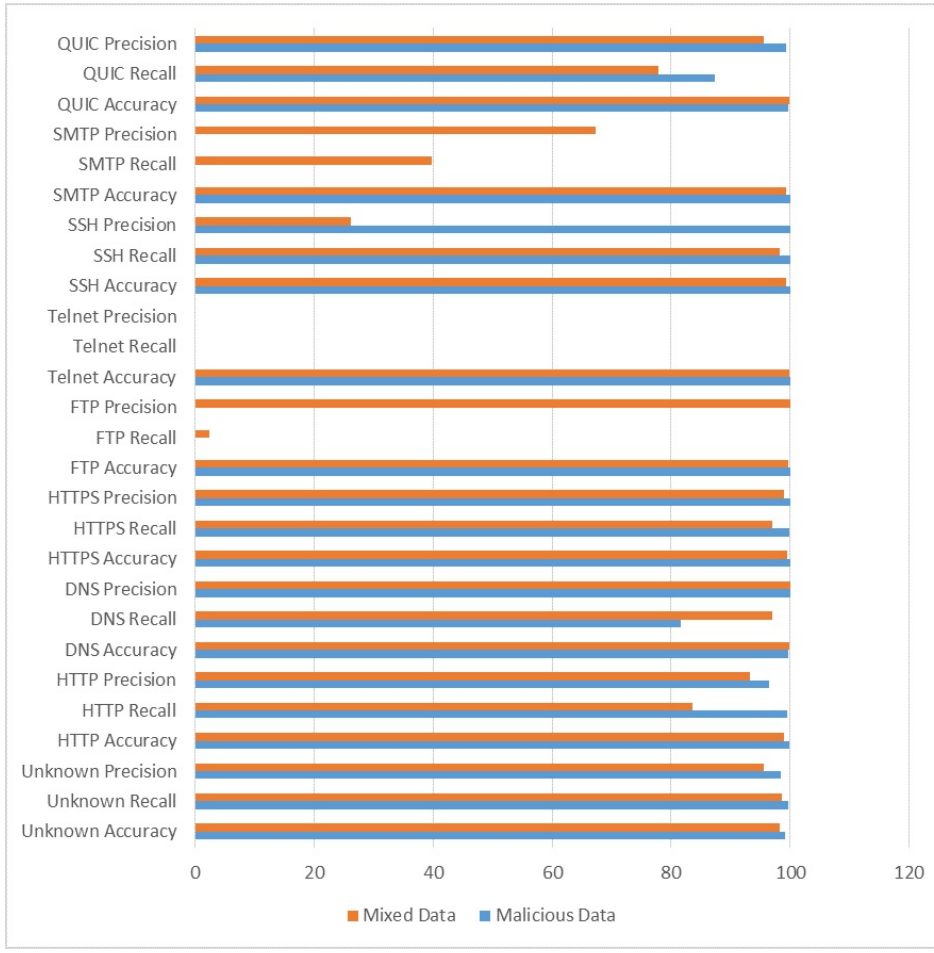


Figure 9. Results for application-specific features based scenario with three packets

Surprisingly, we see a reduction in calculated metrics for different classes compared to the previous tests. Although the reason may be that some valid flows with less than two data packets have been omitted from the results and the amount of true positive results has been reduced, it seems like using two data packets for feature extraction cannot result as good as approaches using fewer data packets. It should also be mentioned that commonly DNS flows only have two data packets consisting of a request packet and a response packet. Also, the existence of HTTP

flows with only one request packet and only one response packet is common. These sentences are mentioned to emphasize the previous statement.

Table 1 provides a numerical representation of all the results. A result of 100 percent for accuracy refers to the fact that all the corresponding flows in test data were correctly classified and no false positive or false negative cases were present. For recall, a result of 100 percent means that there were no false negative cases present and a value of 100 percent for precision means that there were no false positives. For recall and precision, a value of 0 percent means that no true positive cases were present and none of the positive cases were correctly classified.

		Signature-Based		Historical Features		Application Features #1		Application Features #2		Application Features #3	
		Malicious	Mixed	Malicious	Mixed	Malicious	Mixed	Malicious	Mixed	Malicious	Mixed
QUIC	Precision	69.87	75	95.22	94.68	96.53	92.71	99.34	95.74	99.29	95.65
	Recall	100	99.13	91.28	77.39	89.45	77.39	87.72	78.95	87.42	77.88
	Accuracy	98.54	99.92	99.55	99.93	99.53	99.93	99.66	99.94	99.67	99.94
SMTP	Precision	0	99.42	0	97.06	0	68.01	0	72.61	0	67.35
	Recall	0	98.27	0	24.19	0	44.28	0	44.81	0	39.76
	Accuracy	100	99.97	100	98.87	100	99.12	100	99.23	100	99.28
SSH	Precision	100	80.75	100	36.62	75	32.64	100	25.74	100	26.15
	Recall	100	100	100	100	100	96.92	100	98.37	100	98.28
	Accuracy	100	99.93	100	99.51	99.98	99.43	100	99.24	100	99.3
Telnet	Precision	0	98.89	0	0	0	0	0	0	0	0
	Recall	0	100	0	0	0	0	0	0	0	0
	Accuracy	100	100	100	99.81	100	99.81	100	99.81	100	99.81
FTP	Precision	0	100	0	100	0	100	0	100	0	100
	Recall	0	100	0	3.1	0	5.43	0	2.34	0	2.34
	Accuracy	100	100	100	99.73	100	99.74	100	99.73	100	99.73
HTTPS	Precision	56.32	41.09	87.42	42.27	99.93	98.46	99.93	99.07	100	98.97
	Recall	100	100	99.16	99.6	99.16	97.41	99.54	96.68	99.93	97.05
	Accuracy	81.39	78.4	96.37	79.44	99.78	99.38	99.88	99.4	99.98	99.45
DNS	Precision	99.2	87.29	98.22	99.97	99.58	99.97	100	99.99	100	100
	Recall	100	99.98	43.34	85.9	61.84	97.94	95.83	99.67	81.55	97.06
	Accuracy	99.95	96.45	96.58	96.56	97.73	99.49	99.77	99.94	99.7	99.95
HTTP	Precision	97.98	93.8	97.99	84.15	97.5	96.53	98.04	95.05	96.53	93.22
	Recall	99.32	89.76	99.49	92.93	99.66	91.43	99.64	89.22	99.49	83.53
	Accuracy	99.75	98.1	99.77	97.12	99.74	98.62	99.8	98.85	99.88	98.97
Unknown	Precision	99.83	94.59	93.47	82.12	95.37	95.48	98.58	95.9	98.4	95.48
	Recall	64.95	44.94	93.64	52.48	99.34	98.73	99.58	98.73	99.68	98.66
	Accuracy	79.63	72.8	92.27	72.08	96.76	97.17	99.1	98.09	99.24	98.23

Table 1. Results of precision, recall, and accuracy for all scenarios

After discussing the results of traffic prediction in each test for different application classes, it is time to compare different approaches for traffic classification and choosing the most suitable one. The average value for each metric is calculated for all tests and a separate diagram is created for each metric. Figure 10, Figure 11 and Figure 12 present the results of accuracy, recall and precision, respectively.

We can see in Figure 10 that for the malicious traffic, accuracy has been increased by each new test where the lowest accuracy has been experienced in the signature-based test and the highest one is experienced when three data packets are used for feature extraction. The results for the mixed data are similar to the results of malicious traffic as the highest accuracy is achieved when three data packets are used for feature extraction but they have an interesting difference. As we can see, the accuracy has been reduced when using the historical machine learning approach for mixed data. As mentioned previously, the reason for this phenomenon is that the network where malicious traffic has been captured is the same as the network where the training data was captured. Therefore, the flow characteristics may not follow the same pattern historically. It should also be mentioned that the accuracy for all tests using the application-specific features has reached 99 percent and the difference between tests with two data packets and three data packets is negligible.

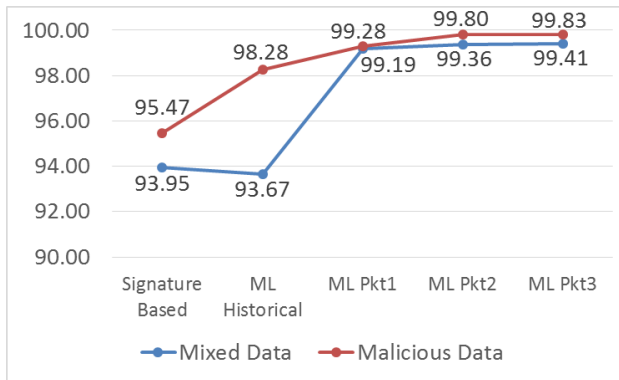


Figure 10. Comparing accuracy for malicious and mixed data

As we can see in Figure 11, the amount of recall is almost the same for both data sets when the signature based approach is utilized at first. For the mixed data we see a reduction in recall value as the tests proceed with the lowest value experienced with historical features approach. For the malicious data, the highest recall value is experienced when two data packets are used for feature extraction and it reduces when the number of packets is increased to three. We believe the reason that the value of recall for malicious data is more than the recall for mixed data is the choice of feature list and the fact that malicious data is captured in the same network as the training data.

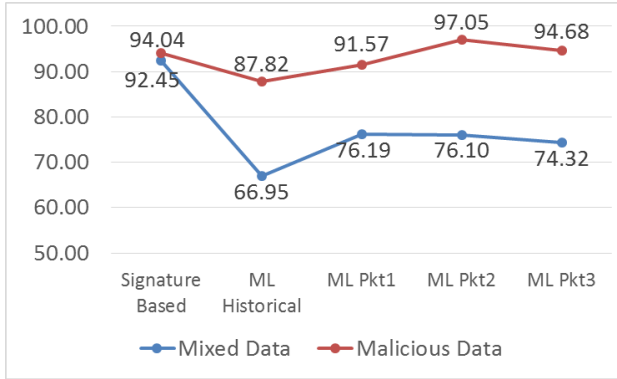


Figure 11. Comparing recall for malicious and mixed data

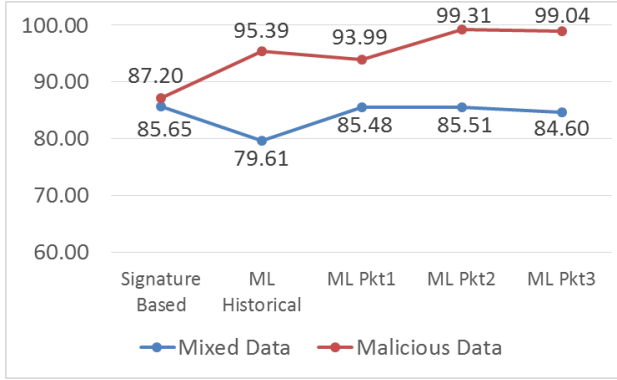


Figure 12. Comparing precision for malicious and mixed data

Like the results presented for recall, the precision value starts almost the same for both data types when the signature-based approach is tested, the lowest value for the mixed data is when historical features machine learning approach is tested and the highest value for the malicious data is seen when two data packets are used for testing the application-specific machine learning approach. Again, it seems like the fact that malicious data is captured in the same network as the training data has had a positive effect on the results.

The reason that signature-based tests for both malicious and mixed data are almost the same is that no feature related to the network environment is used in those tests. Based on the above experiments, we can see that when the training data and the test data are both captured in the same network, we can get the best results with the application-specific features approach when two data packets are used for feature extraction. Considering the case of service function chaining, this theory

may not be the best choice. To make sure, we need to investigate how different approaches affect the load in the network.

Now that we have discussed the classification results in multiple experiments, it is time to discuss about the load imposed on service chains prepared for application classes. When the classification of a flow is done on one of its packets, that packet and the packets after that will all be injected into the chain prepared for the corresponding class. Flows classified in Unknown protocols class are not processed in any chain and their packets are passed unprocessed. If classification is postponed to data packets after the first one, packets in that flow must be processed in all services (in all chains) so the flow receives a complete service and the state created in those services is not incomplete.

Figure 13 presents the amount of load on all service chains in all experiments for mixed and malicious data sets. The load is calculated as the percentage of all data packets (where the actual processing is done) that have entered all service chains over the total number of data packets.

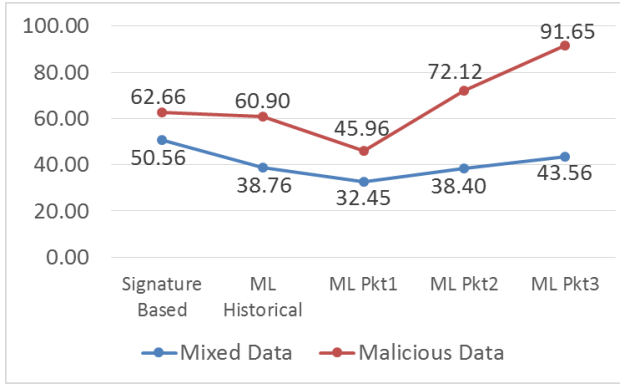


Figure 13. Comparison of load on service chains for mixed and malicious data

What can be seen in the above chart is that as we track the experiments until the one using the application-specific features on the first data packet, the amount of load on all service chains is decreased and gets more meaningful. The reason is that flow classification is becoming more accurate and the resources are not wasted for processing unrelated packets. But as we go on to the experiments using the second and the third data packets, we see an increase in the load imposed on service chains. The reason for this increase is that as the classification is postponed to future data packets, unclassified flows must be processed in all service chains so that the services in the correct service chain do not miss any packets. So, while adding delay to classification time may help in better classification, it results in more load on service chains which is contrary to the main purpose of service chaining idea.

The total trend of the diagram is the same for mixed and malicious traffic data but the load on service chains is higher when testing with the malicious data in contrast with the mixed. The reason for the higher load when using malicious data is not necessarily the result of correctness in classification. A greater proportion of Unknown traffic may be present in the mixed data and when a flow is classified as Unknown, it will not be processed in any of the service chains. So, the variation of traffic classes in a traffic data set can impact the load on service chains.

Before we finish this part and in order to make more powerful results, we have made another comparison between another signature-based classification method and the machine learning methods. This signature-based method is pattern matching. We have labeled the malicious data set with three classes of Unknown, Google, and Instagram and used the same data set in multiple experiments to see how machine learning can be compared with other signature-based methods. Figure 14 presents the results of accuracy, recall, and precision from five experiments using pattern matching and machine learning methods. As we can see, the accuracy is around 95 percent in all the experiments which means that if we consider the correctness of all predictions on network flows, a very good result has been provided. Even if we consider recall, we can see that pattern matching has achieved a value as good as the best machine learning methods.

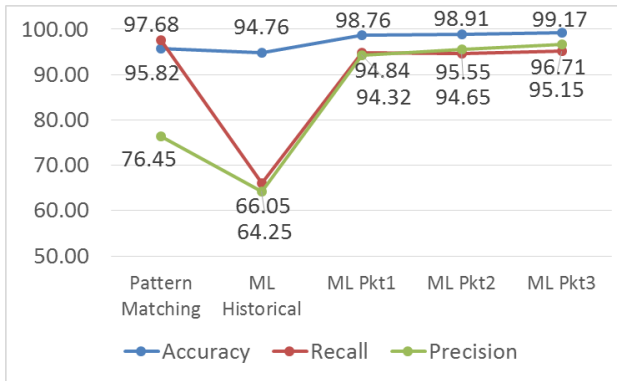


Figure 14. Comparing accuracy, recall and precision when using machine learning methods against pattern matching

The most interesting part of the result is about precision where the pattern matching method presents weaker results compared to some other methods. The DPI module used for labeling the data set leveraged the same pattern matching procedure, but after the application protocol is totally parsed. In other words, the module has intelligently compared each signature pattern with the most appropriate location in packet payload. There are some applications that try to evade detection by using signatures of famous applications in their payload. For example,

a malicious application which uses HTTPS protocol can mention Google in subject alternative name section in its certificate. To avoid the complexities of protocol parsing in DPIs, the pattern matching method used in current experiment has blindly searched for each pattern in the whole packet payload. The reason for a low precision is that there are some flows that have been classified in one of these classes by mistake.

5 CONCLUSIONS

The current work focused on using machine learning techniques in a service function chaining classifier node by leveraging a method named CatBoost. Taking into account the new evasive techniques used by malicious applications active on the Internet, the traditional classification methods like signature-based detection are believed to be ineffective. There needs to be a way for classifying the unknown application flows without following the strict instructions provided by the signatures for known applications. Machine learning can help to classify unknown application flows only by considering the data itself and without the use of application signatures.

By selecting a list of historical and application-specific features and using a labeled data set, we used CatBoost to create models that are used for classifications of unseen network flows in a service function chaining environment. The classified flows were forwarded into a predefined service chain composed of specific services to the selected class where they are processed according to the actual application type. Considering the quality of traffic classification and the load imposed on services, there exists a challenge for selecting the number of packets for extracting the application-specific features. As more packets are considered before the classification of a network flow, more features are available, and possibly, higher quality is achieved.

In our experiments, we have found out that taking into account the service function chaining problem statement, the best number of data packets considered for feature extraction is one. Although selecting one more data packet can slightly increase the classification quality, we believe that the amount of load saved by using only one packet can have more benefits in a service chaining environment. For example, a lower load on service chains can make space for more throughput in the network. This is while the difference in classification quality between methods using one and two data packets is not much at least in respect to accuracy. Also, based on the selection of features extracted from each flow, it is clear that the classification quality will increase if training data and test data are captured in the same network.

Finally, it can be said that using a machine learning method with proper selection of its properties can be a good replacement for traditional signature-based classification and to avoid the expensive, complex procedures of deep packet inspection modules.

REFERENCES

[1] McGregor, A.—Hall, M.—Lorier, P.—Brunskill, J.: Flow Clustering Using Machine Learning Techniques. In: Barakat, C., Pratt, I. (Eds.): Passive and Active Network Measurement (PAM 2004). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3015, 2004, pp. 205–214, doi: 10.1007/978-3-540-24668-8_21.

[2] Moore, A. W.—Zuev, D.: Internet Traffic Classification Using Bayesian Analysis Techniques. 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS '05), Banff, Alberta, Canada, June 2005. ACM SIGMETRICS Performance Evaluation Review, Vol. 33, 2005, No. 1, pp. 50–60, doi: 10.1145/1071690.1064220.

[3] Ertam, F.—Avci, E.: A New Approach for Internet Traffic Classification: GA-WK-ELM. Measurement, Vol. 95, 2017, pp. 135–142, doi: 10.1016/j.measurement.2016.10.001.

[4] Aceto, G.—Ciuonzo, D.—Montieri, A.—Pescapé, A.: Multi-Classification Approaches for Classifying Mobile App Traffic. Journal of Network and Computer Applications, Vol. 103, 2018, pp. 131–145, doi: 10.1016/j.jnca.2017.11.007.

[5] Aceto, G.—Ciuonzo, D.—Montieri, A.—Pescapé, A.: Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges. IEEE Transactions on Network and Service Management, Vol. 16, 2019, No. 2, pp. 445–458, doi: 10.1109/TNSM.2019.2899085.

[6] Lotfollahi, M.—Siavoshani, M. J.—Shirali Hossein Zade, R.—Saberian, M.: Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning. Software Computing, Vol. 24, 2020, No. 3, pp. 1999–2012, doi: 10.1007/s00500-019-04030-2.

[7] Erman, J.—Mahanti, A.—Arlitt, M.: QRP05-4: Internet Traffic Identification Using Machine Learning. IEEE Globecom 2006, San Francisco, CA, USA, 2006, pp. 1–6, doi: 10.1109/GLOCOM.2006.443.

[8] CatBoost Web Site. Available at: `https://tech.yandex.com/catboost/`.

[9] Ghaznavi, M.—Shahriar, N.—Kamali, S.—Ahmed, R.—Boutaba, R.: Distributed Service Function Chaining. IEEE Journal on Selected Areas in Communications, Vol. 35, 2017, No. 11, pp. 2479–2489, doi: 10.1109/JSAC.2017.2760178.

[10] Quinn, P.—Nadeau, T.: Problem Statement for Service Function Chaining. IETF, 2015, available at: `http://tools.ietf.org/html/rfc7498.html`.

[11] Bhamare, D.—Jain, R.—Samaka, M.—Erbad, A.: A Survey on Service Function Chaining. Journal of Network and Computer Applications, Vol. 75, 2016, pp. 138–155, doi: 10.1016/j.jnca.2016.09.001.

[12] Halpern, J.—Pignataro, C.: Service Function Chaining (SFC) Architecture. IETF, 2015, available at: `https://www.rfc-editor.org/rfc/pdfrfc/rfc7665.txt.pdf`.

[13] Li, H.—Wu, Q.—Huang, O.—Boucadair, M. et al.: Service Function Chaining (SFC) Control Plane Components and Requirements. IETF, 2016, available at: `https://tools.ietf.org/pdf/draft-ietf-sfc-control-plane-03.pdf`.

[14] Using Service Classification to Build an Application-Aware NFV Infrastructure for Virtual CPE Services. 2015, available at: https://embedded.communities.intel.com/docs/DOC-8603.

[15] Hantouti, H.—Benamar, N.: Analysis of Service Function Chaining Forwarding Methods, Advances and Drawbacks. The Fourth International Workshop on RFID and Adaptive Wireless Sensor Networks (RAWSN 2016), Marrakesh, Morocco, May 2016.

[16] Zhang, Y.—Beheshti, N.—Beliveau, L.—Lefebvre, G.—Manghirmalani, R.—Mishra, R.—Patneyt, R.—Shirazipour, M.—Subrahmaniam, R.—Truchan, C.—Tatipamula, M.: StEERING: A Software-Defined Networking for Inline Service Chaining. 2013 21st IEEE International Conference on Network Protocols (ICNP), Goettingen, Germany, 2013, pp. 1–10, doi: 10.1109/ICNP.2013.6733615.

[17] Qazi, Z. A.—Tu, C.-C.—Chiang, L.—Miao, R.—Sekar, V.—Yu, M.: SIMPLE-fying Middlebox Policy Enforcement Using SDN. ACM SIGCOMM Computer Communication Review, Vol. 43, 2013, No. 4, pp. 27–38, doi: 10.1145/2534169.2486022.

[18] Jeong, S.—Lee, D.—Hyun, J.—Li, J.—Hong, J. W.-K.: Application-Aware Traffic Engineering in Software-Defined Network. 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), Seoul, South Korea, 2017, pp. 315–318, doi: 10.1109/APNOMS.2017.8094144.

[19] Li, G.—Dong, M.—Ota, K.—Wu, J.—Li, J.—Ye, T.: Deep Packet Inspection Based Application-Aware Traffic Control for Software Defined Networks. 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 2016, pp. 1–6, doi: 10.1109/GLOCOM.2016.7841721.

[20] Li, G.—Zhou, H.—Li, G.—Feng, B.: Application-Aware and Dynamic Security Function Chaining for Mobile Networks. Journal of Internet Services and Information Security (JISIS), Vol. 7, 2017, No. 4, pp. 21–34, doi: 10.22667/JISIS.2017.11.30.021.

[21] Li, G.—Zhou, H.—Feng, B.—Li, G.: Context-Aware Service Function Chaining and Its Cost-Effective Orchestration in Multi-Domain Networks. IEEE Access, Vol. 6, 2018, pp. 34976–34991, doi: 10.1109/ACCESS.2018.2848266.

[22] Bremler-Barr, A.—Harchol, Y.—Hay, D.—Koral, Y.: Deep Packet Inspection as a Service. Proceedings of the 10th ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT '14), Sydney, Australia, 2014, pp. 271–282, doi: 10.1145/2674005.2674984.

[23] Kotsiantis, S. B.: Supervised Machine Learning: A Review of Classification Techniques. Informatica, Vol. 31, 2007, No. 3, pp. 249–268.

[24] Li, W.—Moore, A. W.: A Machine Learning Approach for Efficient Traffic Classification. 2007 15th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, Istanbul, Turkey, 2007, pp. 310–317, doi: 10.1109/MASCOTS.2007.2.

[25] Moore, A. W.—Zuev, D.—Crogan, M.: Discriminators for Use in Flow-Based Classification. September 2005, available at: https://qmro.qmul.ac.uk/xmlui/bitstream/handle/123456789/5050/RR-05-13.pdf.

[26] Jamil, H. A.—Ali, B. M.—Hamdan, M.—Osman, A. E.: Online P2P Internet Traffic Classification and Mitigation Based on Snort and ML. European Journal of Engineering Research and Science, Vol. 4, 2019, No. 10, pp. 131–137, doi: 10.24018/ejers.2019.4.10.1534.

[27] Vlăduţu, A.—Comăneci, D.—Dobre, C.: Internet Traffic Classification Based on Flows' Statistical Properties with Machine Learning. International Journal of Network Management, Vol. 27, 2017, No. 3, Art. No. e1929, 14 pp., doi: 10.1002/nem.1929.

[28] Bakhshi, T.—Ghita, B.: On Internet Traffic Classification: A Two-Phased Machine Learning Approach. Journal of Computer Networks and Communications, Vol. 2016, 2016, Art. No. 2048302, 21 pp., doi: 10.1155/2016/2048302.

[29] Wang, M.—Cui, Y.—Wang, X.—Xiao, S.—Jiang, J.: Machine Learning for Networking: Workflow, Advances and Opportunities. IEEE Network, Vol. 32, 2018, No. 2, pp. 92–99, doi: 10.1109/MNET.2017.1700200.

[30] Bagui, S.—Fang, X.—Kalaimannan, E.—Bagui, S. C.—Sheehan, J.: Comparison of Machine-Learning Algorithms for Classification of VPN Network Traffic Flow Using Time-Related Features. Journal of Cyber Security Technology, Vol. 1, 2017, No. 2, pp. 108–126, doi: 10.1080/23742917.2017.1321891.

[31] Bakhareva, N.—Shukhman, A.—Matveev, A.—Polezhaev, P.—Ushakov, Y.—Legashev, L.: Attack Detection in Enterprise Networks by Machine Learning Methods. 2019 International Russian Automation Conference (RusAutoCon), Sochi, Russia, 2019, pp. 1–6, doi: 10.1109/RUSAUTOCON.2019.8867696.

[32] nDPI: Open and Extensible LGPLv3 Deep Packet Inspection Library. Ntop, available at: https://www.ntop.org/products/deep-packet-inspection/ndpi/.

Habib Allah Khosravi received his B.Sc. degree in information technology engineering from the University of Mazandaran in 2012. He received his M.Sc. degree in computer networks from the University of Isfahan in 2014 and is currently a Ph.D. student in the University of Qom. His research interests include computer networks, network security and distributed systems.

Yaghoub Farjami is Professor of computer science in the University of Qom. He received his Ph.D. degree from the Sharif University of Technology in Tehran, Iran. His research interests include computer security, artificial intelligence and machine learning, cryptocurrencies, etc.

Computing and Informatics, Vol. 39, 2020, 439–463, doi: 10.31577/cai_2020_3_439

TRAFFIC LIGHT RECOGNITION FOR REAL SCENES BASED ON IMAGE PROCESSING AND DEEP LEARNING

Mingliang Che

School of Geographic Science
Nantong University
226019 Nantong, China
e-mail: dawnche@163.com

Mingjun Che

Wuxianshenghuo (Hangzhou) Info Tech Ltd.
311100 Hangzhou, China
e-mail: chemingjun@126.com

Zhenhua Chao, Xinliang Cao

School of Geographic Science
Nantong University
226019 Nantong, China
e-mail: chaozhenhua@ntu.edu.cn, cxliang@mail.ustc.edu.cn

Abstract. Traffic light recognition in urban environments is crucial for vehicle control. Many studies have been devoted to recognizing traffic lights. However, existing recognition methods still face many challenges in terms of accuracy, runtime and size. This paper presents a novel robust traffic light recognition approach that takes into account these three aspects based on image processing and deep learning. The proposed approach adopts a two-stage architecture, first performing detection and then classification. In the detection, the perspective relationship and the fractal dimension are both considered to dramatically reduce the number of invalid candidate boxes, i.e. region proposals. In the classification, the candidate

boxes are classified by SqueezeNet. Finally, the recognized traffic light boxes are reshaped by postprocessing. Compared with several reference models, this approach is significantly competitive in terms of accuracy and runtime. We show that our approach is lightweight, easy to implement, and applicable to smart terminals, mobile devices or embedded devices in practice.

Keywords: Traffic light recognition, color features, perspective relationship, fractal dimension, SqueezeNet

Mathematics Subject Classification 2010: 68U10

1 INTRODUCTION

Traffic light recognition in urban environments is crucial in many cases, such as in detecting traffic signs during semiautomatic or fully autonomous driving [1], helping pedestrians with some form of visual impairment [2], estimating the distance between vehicles and detected traffic lights [3] and assisting in the correction of the map coordinates of a floating car [4]. Thus, many studies address the problem of detecting traffic lights, and it remains an active challenge.

Traffic light recognition belongs to the field of object recognition, i.e. object detection in computer vision. The crucial task of object recognition requires solving two problems: locating objects and then classifying them. Current approaches to traffic light recognition can be divided into two categories: two-stage strategies and one-stage strategies. Regarding the former, the region proposals are generated first. Then, some classifiers are used to classify the region proposals. Many methods can be used to create region proposals. The sliding window method is the easiest, but it is time-consuming. Eventually, selective search was proposed [5, 6]. This method combines the strengths of both an exhaustive search and segmentation. In terms of efficiency, selective search is better than the sliding window method. However, selective search is not sensitive to small objects, such as traffic lights, and its runtime is longer. In contrast to the time-consuming selective search, a speed-up algorithm, i.e. edge detection, was proposed [7]. Edge detection achieves higher object recall and is faster to compute. However, the algorithm does not perform well when segmenting individuals. Another study used binary semantic segmentation to detect region proposals. However, both the precision and computing speed of the method on a CPU need to be improved [8]. Some studies generate region proposals from other perspectives, such as spot light detection [9], color and shape features [10, 11, 12, 13, 14], map information [15] and so on. These methods usually require making strong assumptions and may generate many redundant candidate regions. However, they are based primarily on image processing techniques; thus, their runtimes are relatively short.

In the two-stage strategy, the typical classifiers include template matching [9, 16], the support vector machine (SVM) [17], the hidden Markov model (HMM) [18], and deep learning [17, 19, 20]. The template matching technology is the earliest object recognition application and is easy to operate. However, its rate of correctly recognized traffic lights is closely related to the templates. A stiff template always limits the adaptability of the method to individual objects, especially for dynamic traffic lights in fisheye camera images. The SVM is a better classifier due to its self-learning. However, the SVM performs better on small sample training sets. When the training data becomes large, the SVM becomes time-consuming and has a very limited accuracy when addressing a multiclassification problem. The HMM can help in determining the current state of the traffic light detected based on the obtained state processing. The accuracy achieved by this method is not too high. Owing to the recent advances and performances of deep neural networks, deep learning has been used for image classification. The very representative and popular model is the R-CNN [21] and its accelerated versions [22, 23, 24]. The R-CNN uses the features in the image extracted by deep learning to train the classifier. This process makes the accuracy of object recognition optimal in effect. However, it is time-consuming and requires much storage space. Moreover, it still has problems in identifying small objects, such as traffic lights [25].

In contrast to the two-stage strategy, the one-stage architecture is quite different, as it does not use any prior knowledge of the object locations. It uses algorithms to directly output categories and the corresponding positioning. The more popular models include YOLO [26] and the SSD [27]. YOLO, that is, "You Only Look Once", is a unified and real-time object detector that uses only a single deep neural network to detect objects in images. The smaller version of YOLO, i.e. fast YOLO, performs well when conducting real-time object detection in a video [28]. However, YOLO makes more localization errors and lags behind state-of-the-art detection systems in terms of accuracy [26]. Thus, an improved YOLO approach, i.e. YOLO v2 (also called YOLO 9000), was proposed. Compared with YOLO, YOLO v2 uses a novel multiscale training method, runs at varying sizes, and offers an easy tradeoff between speed and accuracy [29]. Then, an advanced YOLO approach, i.e. YOLO v3, was presented. YOLO v3 is obviously faster than the reference approaches when realizing the same accuracy [30]. From YOLO to YOLO v3, the speed increases, while the accuracy does not, especially regarding the location precision. The SSD model appeared after the YOLO model. The SSD completely eliminates proposal generation and subsequent pixel or feature resampling stages and encapsulates all computations in a single network, which makes the SSD easy to train and optimize. The experimental results show that the SSD has competitive accuracy relative to that of the comparison models and is much faster. However, the SSD is not very good at or even unable to perform small object recognition. In addition, similar to the YOLO model, the size of the SSD is also very large.

Overall, the two-stage approaches have higher precision and a longer runtime, while the one-stage methods have lower precision but a shorter runtime. Deep

learning has high precision, but its model is usually very large, which is extremely disadvantageous for devices with lower memory and weaker processors outside the laboratory. Moreover, deep learning does not seem to be compatible with the accuracy and runtime.

In this paper, our approach takes a two-stage architecture to ensure high recognition accuracy. We used image processing to generate the region proposals and then applied the deep convolution neural network (CNN) to classify them. To reduce the runtime, we further optimized the detector by considering the perspective relationship and the fractal dimension. To decrease the model size, we introduced the lightweight SqueezeNet model, which strongly compresses the dimensions of the feature map [31]. Finally, based on the ground-truth image dataset, the performance of our approach was tested.

2 MATERIALS AND METHODS

2.1 Dataset Description

To evaluate our traffic signal recognition approach, two datasets for real scenes were employed. The first dataset includes data that we collected, which were derived from the position-image synchronous data captured by the driving recorder on a floating car. The time interval of the image set is 5 s. The dataset was recorded in the urban areas of Nantong and contained red, green and blue (RGB) color images. The image quality of the dataset is relatively high and has a resolution of 800×600 pixels. The dataset contains more than 700 images and more than 1 000 traffic lights. The scenes in the dataset can be divided into simple scenes and complex scenes. In the former, the traffic signals are exposed to the sky and are easily identifiable. In the latter, the backgrounds of traffic signals are diverse and may include trees, buildings, piers, and billboards. Moreover, the taillights of cars and outdoor lights resemble traffic signals from a distance, which causes further interference. In this situation, traffic signals are not recognizable.

The latter dataset is the publicly available benchmark dataset of the La Route Automatise (LaRA) benchmark provided by de Charette et al. [9]. The camera applied in the LaRA dataset has a focal length of 12 mm and a resolution of 640×480 pixels. The image color mode of the dataset is RGB full color. This dataset has been recorded in French urban areas and has more than 11 000 frames, many of which are stationary. To strengthen the difference between two adjacent frames, we resampled the source dataset every 5 frames. In the resampled data, some very blurry frames will be removed manually. The final resampled data contain more than 1 500 frames and more than 2 000 traffic lights. Use of the dataset allows a comparison between our approach and the reference methods.

2.2 Method Schema

2.2.1 Detector

A traffic light has a few special and important features, such as its border shape, lamp holder color and light color, which help people to distinguish it in real scenes. However, in image processing, the traffic light border shape is difficult to extract because it is easily affected by the illumination intensity, the light size and disturbances from background pixels. At night or when the traffic light is either very small or embedded in foliage, the traffic light border shape is very difficult to estimate. The illumination intensity and the light size also significantly affect the lamp holder color in the image. By contrast, the traffic light color, which is noteworthy and unique whether at day or at night, is relatively stable and is applied to locate the candidate regions of traffic lights.

The images in RGB mode are first converted to the hue, saturation, and value (HSV) color space. Compared with the RGB color space, the HSV color space is more suitable for segmentation and is more robust against illumination variation [12]. In this section, the regions of red, green and yellow are extracted to create the candidate boxes, i.e. the region proposals, that contain the traffic lights. The desired color is extracted from an image based on the HSV values resulting from the statistics of the positive samplings (Figure 1).

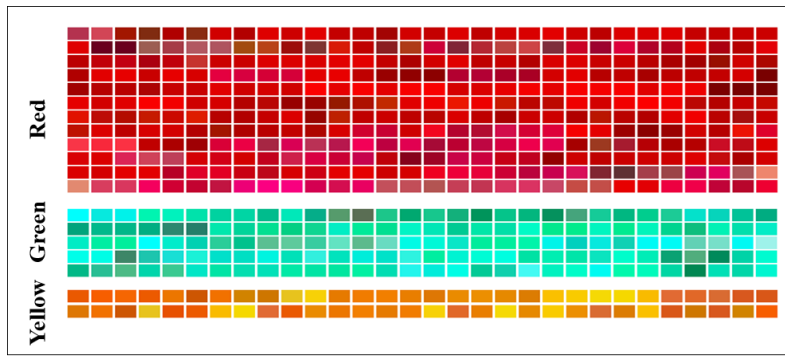


Figure 1. Color statistics of traffic lights in RGB mode

Suppose the triple (h, s, v) represents the HSV value of every pixel in an image; the expected color needs to satisfy the empirical relationships. In Figure 1, the range of each component in the color triple is calculated. Then, there is a slight change in the range. This process is performed to cover the traffic light color as much as possible and to find as many candidate regions as possible (Figure 2 b)).

To detect the red lights, the color thresholds were set as follows.

$$0 \leq h_{red1} < 15, \quad 115 \leq s_{red1} \leq 255, \quad 115 \leq v_{red1} \leq 255, \tag{1}$$

$$165 \leq h_{red2} \leq 180, \quad 120 \leq s_{red2} \leq 255, \quad 90 \leq v_{red2} \leq 255. \tag{2}$$

To extract the green and yellow lights, the color thresholds were set as shown below.

$$55 \leq h_{green} < 90, \quad 60 \leq s_{green} \leq 255, \quad 90 \leq v_{green} \leq 255, \tag{3}$$

$$15 \leq h_{yellow} \leq 25, \quad 195 \leq s_{yellow} \leq 255, \quad 205 \leq v_{yellow} \leq 255. \tag{4}$$

During the color extraction, the candidate regions, i.e. the approximate positions of the traffic lights, were detected (Figure 2 b)), and the corresponding binary images were created. To further calculate the profiles of the candidate regions, the *CHAIN_APPROX_SIMPLE* contour approximation algorithm in OpenCV was applied. In the algorithm, by compressing the horizontal, vertical, and diagonal segments and leaving only the end points, as little key pixels as possible are used to present the outlines. According to the contours of the candidate regions, the homologous bounding rectangle can be calculated. Then, the candidate box was estimated as follows.

$$bw = \text{int}(w' \cdot c \cdot k), \quad bh = \text{int}(h' \cdot c \cdot k), \tag{5}$$

$$w' = \min(w, h), \quad h' = \min(w, h) \tag{6}$$

where bw and bh are the width and height of the candidate box, respectively. w and h are the width and height of the bounding rectangle, respectively. The coefficient c is the ratio of the traffic light width to the lamp holder border width. In this paper, c equals 2.0. The zoom factor k ranges from 1.0 to 1.5. It can expand the background pixel information around the traffic light and can contribute to determining whether or not the candidate region contains the traffic light.

In the candidate boxes, some contain traffic lights, while some contain noise (Figure 2 c)). Apparently, some candidate boxes do not contain traffic lights because their size is inappropriate. To remove the obvious noise, the size estimation formula for the traffic lights was used. Depending on the perspective, everything looks small at a distance and large up close. Thus, a mathematical relationship exists between the light width and the corresponding y-coordinate value (Figure 3).

$$bw' = -0.101 \cdot y + 38.43 + t \tag{7}$$

where bw' denotes the estimated box width, y denotes the y-coordinate value, and t denotes the tolerance. When bw is less than bw', the candidate box is most likely noise. With this process, noisy boxes that are obviously smaller or larger than the normal size will be removed (Figure 2 d)). At the same time, some indistinguishable disturbance terms, e.g., some countdowns of traffic lights, are also removed (Figure 2 d)). Obviously, this process is carried out to reduce the number of false positives and shorten the runtime for recognizing traffic lights.

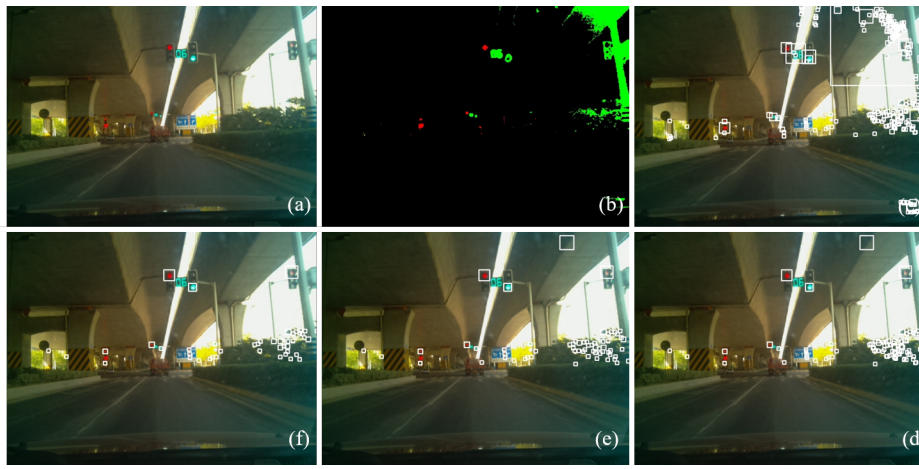


Figure 2. Sample detections of traffic lights in an image. a) is the original image, b) represents the candidate regions, c) represents the rough candidate boxes, d) represents the candidate boxes processed via size-noise reduction based on the perspective relationship, e) represents the candidate boxes processed via overlap removal with IOU, and f) represents the candidate boxes further processed via texture-noise reduction with the fractal dimension.

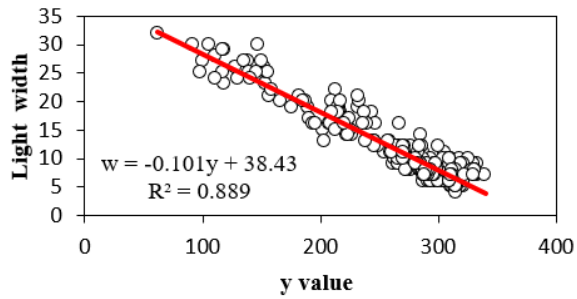


Figure 3. Relationship between the light width and the corresponding y-coordinate value

During the image processing above, multiple candidate boxes may be covering each other around the same traffic light. To remove the overlapping boxes, the intersection over union (IOU) method was applied, and the threshold was set to 0.6. As a result, the remaining candidate boxes became small in number but contained nearly all traffic lights (Figure 2 e)).

Now, the detector is quite qualified for the detection task. To further shorten the runtime of the proposed approach, we need to reduce the backgrounds of the candidate boxes. In Figure 2 e), some candidate boxes do not visibly contain traffic lights. We can make this judgment by the image texture. The fractal dimension

is a useful feature for texture segmentation in image processing [32]. The box-counting approach is one of the most frequently used techniques for estimating the fractal dimension of an image and is defined below.

$$F = \lim_{r \to 0} \frac{\log(N_r)}{\log(1/r)} \tag{8}$$

where F is the fractal dimension and N_r is the least number of boxes that must completely cover the broken lines in the scale r. Before calculating the fractal dimension, the image first needs to be processed by the Sobel filter. Then, the image is transformed to binarization. Thus, the gray plane is converted to the broken lines in two dimensions (Figure 4). By calculating the numbers of white and black pixels, respectively, the ratio equation is as follows.

$$R = \frac{N(pix = 255)}{N(pix = 255) + N(pix = 0)} \tag{9}$$

where R is the ratio and $N(pix = 255)$ and $N(pix = 0)$ denote the number of pixels, with values equal to 255 and 0, respectively.

The conspicuous texture discrepancy between traffic lights and backgrounds can be quantified by calculating the fractal dimension and the R value (Figure 4). The fractal dimension of traffic lights principally ranges from 1.3 to 2.0. However, the fractal dimension range for backgrounds is from 0 to 2. Thus, a threshold line can be drawn to distinguish them. The R value is also helpful in distinguishing traffic lights and backgrounds. As shown in Figure 4 g), the fractal dimension of the image was high, indicating that the texture of the image was rough. The spatial distribution of the texture may be similar to that of traffic lights, which makes it difficult to distinguish the texture. However, its R value was evidently lower than that of traffic lights. Therefore, the texture noise can be further eliminated by the R value.

2.2.2 Classifier

Once the candidate boxes are generated, the classification task can begin. In this paper, the candidate boxes contain four states, i.e. the backgrounds, red lights, green lights and yellow lights. To distinguish them, SqueezeNet was employed. Among the SqueezeNet modules, the fire module, which is comprised of a squeeze convolution layer and an expand layer, is the building block. It can enable the CNN to preserve accuracy on a limited budget of parameters [31]. SqueezeNet begins with a standalone convolution layer (conv1), followed by 8 fire modules (fire2–9), and finally ending with a conv layer (conv10). The number of filters per fire module from the beginning to the end is gradually increased. Moreover, SqueezeNet performs max-pooling with a stride of 2 after layers conv1, fire4, fire8, and conv10. According to the SqueezeNet architecture, we reproduced it with the TensorFlow library. The size of the input image was adjusted to 64×64. The channel of each layer retained

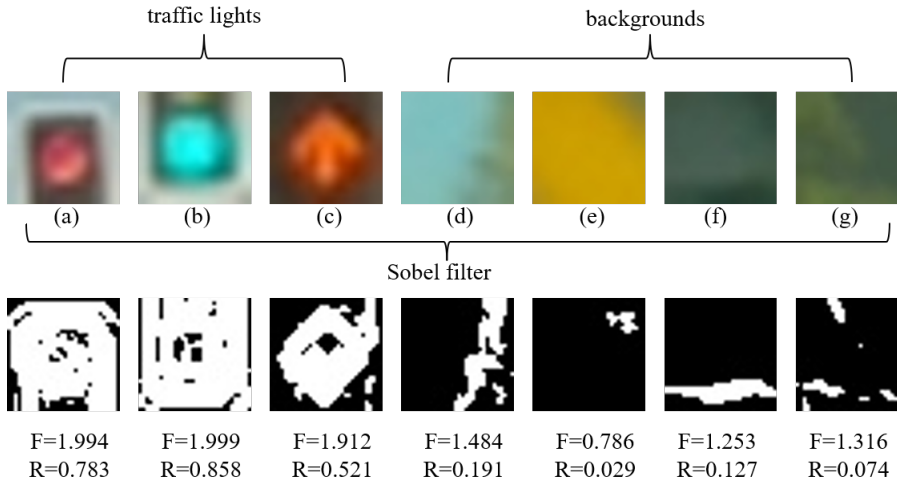


Figure 4. Texture discrepancy between traffic lights and backgrounds

its original value. The number of categories was set to 4. The learning rate was set to 0.0001. The input images were first processed by the equalization method before training SqueezeNet.

2.2.3 Postprocessing

The candidate boxes were transformed into classified boxes after processing them with the classifier. At this step, several overlaps may still exist among the boxes. To collapse them, the non-maximum suppression (NMS) method was needed. NMS is a key postprocessing step in many computer vision tasks. On the basis of the sorted scores, it uses an iterative procedure to retain only one box per group, corresponding to the precise local maximum of the response function, ideally obtaining only one detection per object [33]. Then, the perspective relationship (Equation (7)) was used again to slightly reshape the boundaries of the recognized traffic lights. The overall processes of the entire method are summarized in Figure 5.

2.3 Reference Methods

The performance of the proposed approach is compared with that of several popular approaches. One approach is YOLO v3, which was briefly introduced in the first section. YOLO v3 still imitates the mechanism of the human visual system. It strives to predict what objects are present and where they are by looking at an image only once. Similar to the previous versions, YOLO v3 regards object detection as a single regression problem, straight from image pixels to bounding box coordinates

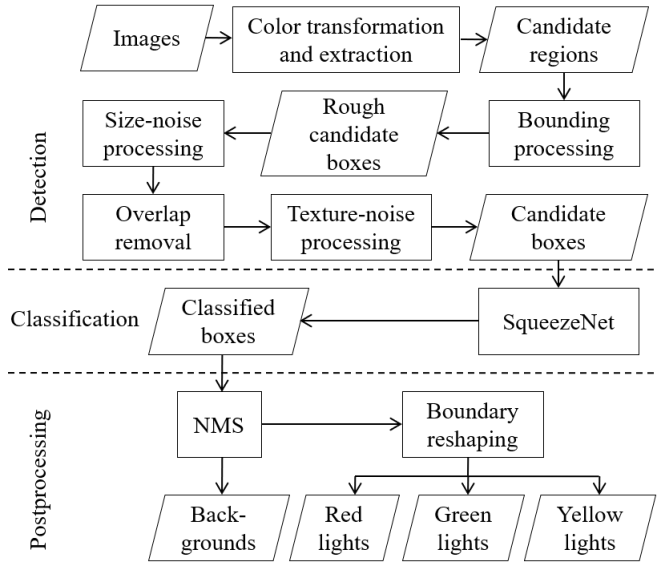


Figure 5. Flow chart of the proposed approach

and class probabilities. The codes of YOLO are open. According to the instructions, YOLO v3 is used on the Linux platform.

Other approaches include the de Charette approach [9], DeepTLR model [19], FusionTLR model [25], Bayesian algorithm [35], Haltakov approach [11], and Siogkas approach [36]. These approaches, which have been validated using a publicly available dataset, are applied to evaluate our method. Details about these approaches are provided in the corresponding literature.

2.4 Evaluation Metrics

The approaches used to evaluate the recognition performances are defined as follows:

$$Acc = (TP + TN)/(TP + TN + FP + FN), \tag{10}$$

$$Pre = TP/(TP + FP), \tag{11}$$

$$Rec = TP/(TP + FN), \tag{12}$$

$$F1 = 2 \cdot (Pre \times Rec)/(Pre + Rec) \tag{13}$$

where Acc is the accuracy, TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives. Pre is the precision, and Rec is the recall. According to the Pre and Rec values, the $F1$ score is calculated.

When compared with other models, the metric 'AUC', i.e. the area-under-the-curve, of the precision-recall curves was also used in addition to the above indicators.

2.5 Experimental Situation

The image dataset was divided into two parts: training data and verification data. In the training data, the training samples created by the detector contain the red lights, green lights, yellow lights and backgrounds. The backgrounds account for nearly 94 %. To balance the sample ratios, we used some strategies to process the positive sample data. These strategies include moving the four borders of the traffic light separately, graying the image, scaling the image, using a Gaussian filter, sharpening the image, equalizing the image, stretching the image to the left and right, and stretching the image upward and downward. Based on the image dataset, the model training and verification platform was a single-core CPU @ 2.5 GHz.

3 RESULTS ANALYSIS

3.1 Classifier Performance

The classifier, i.e. SqueezeNet, used in the proposed approach was trained 25 000 times by learning the training datasets (Figure 6). The corresponding weight files were produced per 2 500 times. The accuracy varied sharply before 3 000 iterations and gently after. The maximum value of the accuracy was 0.973, which appeared near the 22 000$^{\text{th}}$ iteration. The variation in the loss function curve was acute from beginning to end. However, the trend of the loss curve was clear. At the 12 500$^{\text{th}}$ iteration, the minimum value appeared, and the corresponding accuracy was 0.968. After that, the change was slightly dramatic. However, the trend of the loss curve was not affected.

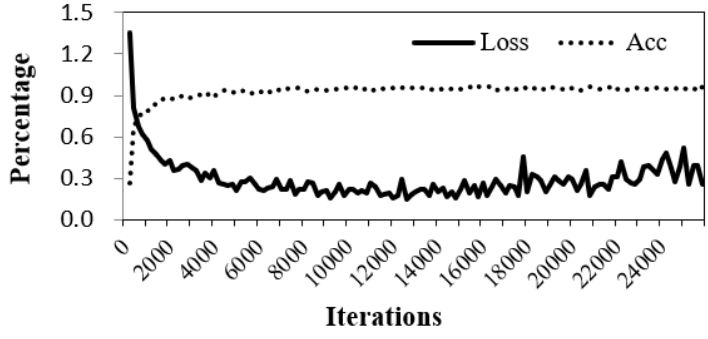


Figure 6. Training curve of the classifier

The validation dataset was used to evaluate the classification performance. Figure 7 shows the precision and recall of the classifier at different iterations. Before the

$7\,500^{\text{th}}$ iteration, the manifestation, i.e. distinguishing the four types of candidate boxes, was unsatisfactory. In this stage, the classifier distinguished the red traffic lights and the green traffic lights comparatively easily. However, for the yellow traffic lights and the backgrounds the situation was terrible. The precision of distinguishing backgrounds was lower than 0.8, and the recall was higher than 0.9. The recall of distinguishing the yellow traffic lights was lower than 0.7, and the precision was higher than 0.87. These results indicate that more yellow traffic lights were falsely classified as backgrounds by the classifier. After that the $7\,500^{\text{th}}$ iteration, the classifier produced better classification results. Both the precision and the recall when distinguishing the four types of candidate boxes were high. Figure 8 also provides the overall classification accuracy and the corresponding *F1* score. Obviously, at the $12\,500^{\text{th}}$ iteration, the performance of the classifier was the best. The performance when distinguishing red traffic lights was inferior to that when distinguishing green traffic lights but better than that when distinguishing yellow traffic lights.

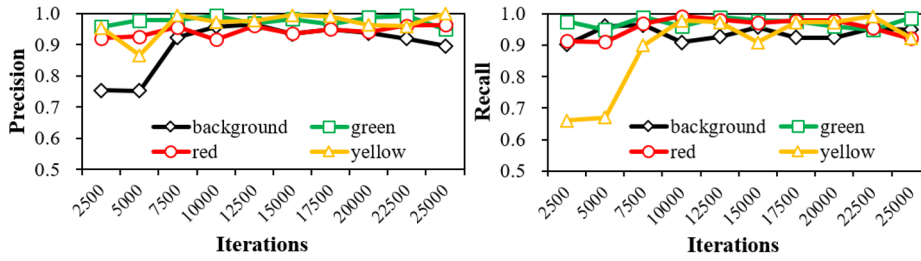


Figure 7. Precision a) and recall b) of the classifier

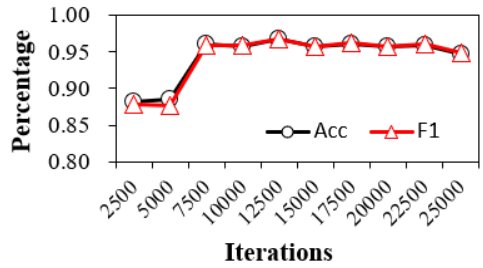


Figure 8. Overall classification accuracy and *F1* score of the classifier

3.2 Recognition Evaluation

The verification images were used to evaluate the recognition performance of the proposed approach. The *F1* score varied with the IOU threshold (Figure 9). Obviously, the overall recognition accuracy declined with increasing IOU. The situation

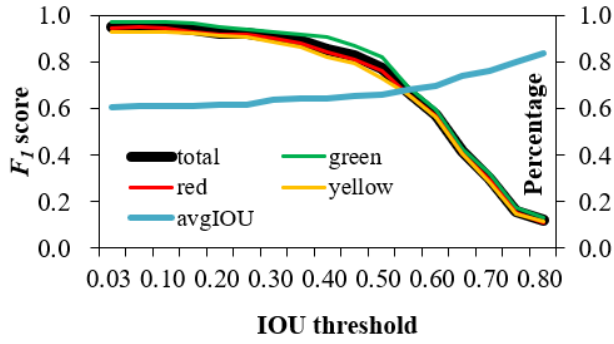


Figure 9. Relationships between the recognition accuracy and the IOU

remained the same for each type of traffic light. However, irrespective of the change in the IOU, the average IOU curve always exceeded 0.6. When the IOU was lower than 0.1, the total *F1* score maximum value was 0.951, and the corresponding average IOU was 0.611. In the figure, when the IOU falls between 0.1 and 0.3, the total *F1* score only slightly varied and was always higher than 0.9. When the IOU varied from 0.3 to 0.45, the total *F1* score was always greater than 0.8. When the IOU was between 0.45 and 0.6, the total *F1* score was always greater than 0.6.

These results suggest that the performance of the proposed approach in terms of recognizing traffic lights was satisfactory. The estimation of the light boundaries by the proposed approach was also approbatory. Moreover, the red traffic light recognition performance achieved by the proposed approach was inferior to the performance of green traffic light recognition but superior to that of yellow traffic light recognition.

The runtime of the proposed approach was analyzed (Figure 10). After the addition or removal of modules, the runtimes of the corresponding approaches were compared. In Figure 10 a), the total runtime of the 'full' approach was the shortest, followed by that of the 'perspective' approach, while the longest was that of the 'none' approach. In the runtime list of each approach, the elapsed time of the detector was the shortest. However, the detector of the 'fractal' approach had a longer elapsed time than that of the other approaches, which was associated with the fractal computation. Relative to the others, the detector of the 'fractal' approach required extra time to accomplish the fractal-computation task. Multiple values needed to be calculated in each candidate box to estimate the fractal dimension. Thus, the stage of the 'fractal' approach was somewhat time-consuming.

The elapsed time of the classifier was longest and directly affected the total runtime of each approach. In fact, in this stage, the elapsed time was related to the classifier-self and the number of candidate boxes. The former, i.e., the classifier distinguishing each candidate box, was approximately 5.5 ms. The latter was not constant. When the number of candidate boxes was large, the elapsed time of the

classifier was necessarily long, and vice versa. When neither the perspective relation module nor the fractal dimension module was used, the number of candidate boxes was very large. Conversely, the number of candidate boxes was small. Thus, the elapsed time of the 'none' approach was the longest, followed by that of the 'fractal' approach, while the shortest was that of the 'all' approach. Therefore, the two modules played important roles in reducing the runtime (Figure 10 b)). The time compression ratio of the perspective relation module was as high as 0.57 and that of the fractal dimension module was 0.29. Using the two modules synchronously, the time compression ratio was as high as 0.62. Finally, the average runtime needed to process a single image for the 'full' approach was approximately 240 ms, indicating that the proposed approach can process four images per second. When detecting traffic lights under the offline condition, the proposed approach was very fast. Even so, the proposed approach can completely satisfy the practical requests for some online tasks requiring real-time image processing, such as processing the floating car data from the low frequency (> 30 s) to the high frequency (1–10 s).

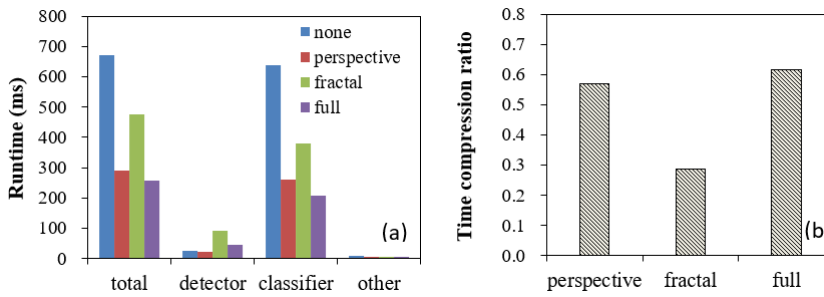


Figure 10. Analysis of runtime. In the figures, 'perspective' denotes the approach containing the perspective relation module; 'fractal' denotes the approach containing the fractal dimension module; 'all' denotes the approach containing the two modules; and 'none' represents the approach without the two modules.

Figure 11 shows the performance in recognizing the different sizes of traffic lights. The minimum resolution of the proposed approach for recognizing the traffic lights was set to five pixels. When the size of the traffic lights was lower than 8 pixels, the *F1* score was always higher than 0.92. When the size of the traffic lights was expanded to 11 pixels, the proposed approach yielded its best results. Then, the *F1* score of the proposed approach declined. This decline is related to the chromatism of traffic lights, which is frequently caused by light pollution. For example, when the backlight behind the traffic light is too bright, the traffic light color becomes lighter. In this case, the proposed approach can do nothing due to the existence of chromatism. The statistics showed that larger traffic lights were more likely to be impacted by light pollution than smaller lights, which explains the phenomenon above. From another perspective, the proposed approach has prominent advantages in recognizing small traffic lights.

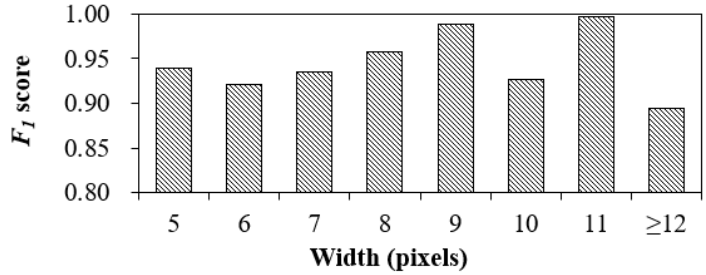


Figure 11. Traffic light size recognition performance achieved by using the proposed approach

4 DISCUSSION

4.1 Comparison with the Other Reference Models

A comparison of the recognition results of different approaches based on the LaRA dataset is shown in Table 1. Compared with state-of-the-art approaches, the *F1* score of the proposed approach was slightly lower than that of the Bayesian algorithm but higher than the scores of other approaches. The recognition performance of the proposed approach in the reference models is superior. Similar to the Bayesian algorithm and de Charette approach, the precision of the proposed approach was higher than that of the recall. This finding shows that the error rate for traffic signals recognized by the proposed approach was lower than the omission rate. However, opposite results were obtained for the Haltakov approach, Siogkas approach and DeepTLR model. During the verification, we discovered that some traffic signals appeared very small and some light textures were vague. In addition, the similarity between the traffic signals and the backgrounds due to the lower image quality hindered detection. These factors reduced the recall of the proposed approach and increased the omission rate. In addition to comparing the *F1* scores, the area under the curve (AUC) indicators were also compared. Specifically, a comparison between the proposed approach and the FusionTLR model was performed. The red light recognition performance achieved by the FusionTLR model was obviously better than the green light recognition performance. However, the case of the proposed approach was completely different. For the LaRA dataset, some red signals that were quite small and blurry were very similar to the backgrounds. Thus, their detection using the proposed approach was difficult.

The runtime costs were also analyzed. The runtime of the proposed approach was approximately 120 ms per image on the LaRA dataset, which is longer than that of most approaches and indicates that the proposed approach based on the current implementation was relatively disadvantaged in processing the video images.

Approach	Precision	Recall	*F1*	$\mathbf{AUC}_{Red}$	$\mathbf{AUC}_{Green}$	Platform	Reference
Bayesian algo.	0.987	0.947	0.966				[35]
FusionTLR				0.920	0.893	CPU	[25]
Haltakov approach	0.728	0.801	0.763			CPU	[11]
Siogkas approach	0.612	0.938	0.741			CPU	[36]
DeepTLR	0.856	0.907	0.881			GPU	[19]
de Charette approach	0.845	0.535	0.655			CPU	[9]
Ours	0.904	0.897	0.900	0.898	0.930	CPU	

Table 1. Comparison of the proposed approach with state-of-the-art methods using the LaRA dataset

Furthermore, we compared the proposed approach with the other more famous object detection model, i.e. the YOLO model. The validation results revealed that the performances of YOLO were unsatisfactory on both our datasets and the LaRA dataset. The statistical results showed that both the precision and recall of YOLO were lower than those of the proposed approach. A comparison of their vision results is shown in Figure 12. Subplot (a) shows that the proposed approach can distinguish the red and yellow lights better. However, YOLO falsely classified the yellow lights as red lights. In subplot (b2), mistakes and omissions in recognizing the traffic lights occurred when using YOLO. This observation indicates that YOLO is unsuitable for detecting small objects, which is in agreement with the viewpoints of some studies [20, 25].

4.2 Strategies for Further Model Optimization

4.2.1 Improving the Recognition Accuracy

The recognition accuracy of the proposed approach depends on the candidate box quality of the detector and the classification accuracy of the classifier. The candidate boxes produced by our detector did not ensure coverage of all traffic lights. The statistics showed that the probability of the candidate boxes covering all traffic lights was approximately 0.98. The reason for this was that our detector was sensitive to the image color, which was closely related to the image quality. Many factors affect the image quality of an on-vehicle camera. The major elements contain image blurring due to shaking of the camera, color distortion due to light pollution, and the lower resolution of the camera. Upon encountering poor image quality, omission may occur for the candidate boxes. Therefore, further optimization of our detector will be studied in future work.

In the subsequent classification, a deep CNN, i.e. SqueezeNet, was applied. SqueezeNet is small and has a high classification precision. However, in this paper,

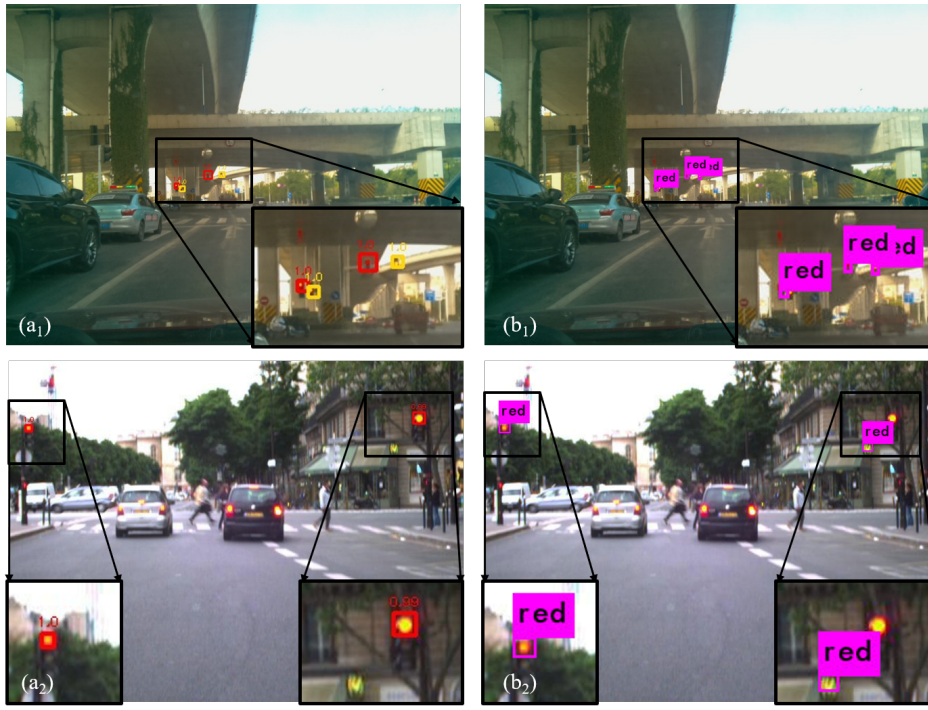


Figure 12. Vision results on 1) our validation dataset and 2) the LaRA validation dataset obtained by a) the proposed approach and b) the YOLO model

there is an upper limit to its classification precision. When training SqueezeNet, we found some difficulty in exceeding the *F1* score of 0.97 regardless of how the training samples were adjusted. Thus, we designed and developed a new and simple but highly efficient deep CNN classifier, denoted as SimpleNet. SimpleNet contains two convolution layers, two pooling layers and two fully connected layers. The loss function uses cross entropy. After tens of thousands of iterations, the *F1* score of SimpleNet can reach 0.985, which is then very difficult to exceed. Figure 13 shows the misclassifications between the classifiers. Obviously, both classifiers faced the same major challenges in distinguishing the backgrounds and traffic lights. However, the misclassification error between the interiors of the traffic lights was lower for the classifiers. Although SimpleNet has better classification performance, its file size is large, and the runtime is long. Therefore, some tradeoff among the model runtime, accuracy and size must occur.

By examining the classification data, we found that the contexts around some traffic lights were inadequate. This inadequacy caused the backgrounds to be similar to the traffic lights, especially for small traffic lights. The smallest resolution of the proposed approach was 5 pixels. When the size further decreased, the traffic

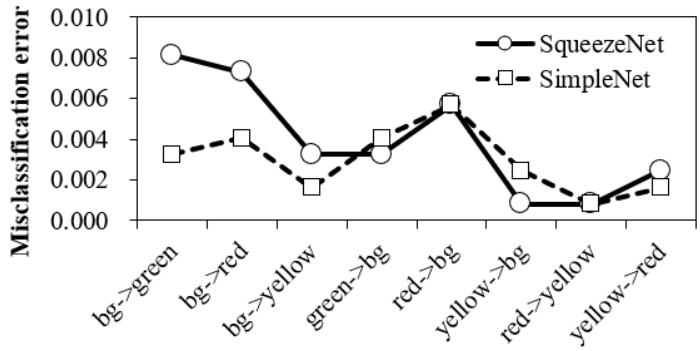


Figure 13. Analysis of the misclassification of classifiers

lights became very similar in appearance to the backgrounds. As a result, it was difficult to detect them by human eyes. This situation makes some traffic light detection systems meaningless because the on-vehicle camera was too far away from the traffic lights. Usually, human eyes require an increase in context information, i.e., expansion of the visual boundaries, to recognize small objects. However, this increase did not significantly improve the classifier performance because too many contextual data will overwrite the features of traffic lights. This deficiency stems from the fact that the classification by the CNN is based on the local image texture [34]. Thus, to resolve this problem, the use of contextual data to improve the classification performance of our classifier will be researched in future work.

4.2.2 Reducing the Runtime

Compared with that of the other traffic light recognition approaches, the runtime of the proposed method is slightly longer. However, the runtime of the proposed method can be considerably reduced in theory.

First, the proposed approach adopts serial computing technology, i.e. single-thread sequential processing, in the implementation. In the candidate box classification, the classifier sequentially classifies the candidate boxes. If more candidate boxes exist, the serial processing mechanism of the classifier will lead to accumulation of the runtime. Therefore, the overall runtime will be longer. In fact, no correlation among the candidate boxes exists, which is good for parallel computing. Under a single-core CPU, the processing of a single candidate box with our classifier requires approximately 5.5 ms. If multiprocess is used, according to Figure 10 a), the total time needed by the proposed approach to process a single image should not exceed 25 ms to ensure that video image processing and online real-time processing are possible. Therefore, in theory, it is feasible to use parallel computing to optimize the proposed approach, and the computational efficiency after the optimization will be significantly improved.

Second, for video images, the target tracker technology, which has a runtime advantage when processing frame images, can be coupled to our model. The multiple trackers were initialized at first by the traffic light boxes recognized by the proposed approach in the previous frame image and were then used to find the new positions of traffic lights in the next frame image and update the target position corresponding to each tracker, thereby improving the traffic light detection efficiency.

Finally, for on-vehicle camera images, the location of traffic lights has a certain regularity. In some areas, a high probability of traffic lights exists, while in other areas, the probability might be small or equal to zero. For example, the probability of a traffic light appearing above the horizon in an image is large. In contrast, traffic lights typically do not appear below the horizon. Therefore, an invalid scan of the algorithm can be prevented by setting the view boundary appropriately, thereby improving the detection efficiency and quality.

In addition, using a GPU for calculations may also significantly improve the efficiency of the proposed approach. Therefore, according to the above strategies, further optimization of the proposed approach to reduce the runtime will be performed in future studies.

4.3 Potential for Migrating to Embedded Devices

Currently, automatic recognition of traffic signs is very important for vehicle automatic driving or assisted driving. Traffic sign recognition systems have become an integral part of Advanced Driver Assistance Systems (ADAS). Fast, robust and real-time automatic traffic sign detection and recognition can significantly increase driving safety. Although many traffic signal recognition methods are available, they are not easily migrated to embedded devices because the hardware performance of embedded devices is generally lower than the computer conditions in the laboratory.

With the constraint, the proposed approach should meet the specified conditions. First, the proposed approach has a high accuracy rate and recall rate in terms of the recognition performance and has better performance in reference approaches. Second, the proposed approach has the potential to improve the time efficiency in terms of the cost calculation. According to the previous analysis, when multiple processes are employed for parallel classification and the target tracker technology has been adopted, this approach is fully applicable to online real-time recognition tasks. Last, the proposed approach is lightweight. The total size of the approach, including the trained weight file, does not exceed 10 MB. The approach only depends on the OpenCV and Tensorflow libraries during implementation; thus, it has minimal dependence on third-party development libraries and implementation is simple. These characteristics enable the proposed approach to have great potential for migration to embedded devices.

Similar to other methods, the proposed approach inevitably contains some parameters and coefficients, such as the hue, saturation and value (HSV) thresholds

applied to extract the traffic signal candidate regions, the coefficients for establishing the perspective relationship equation, and the fractal and R thresholds applied for texture denoising. The optimal values of these parameters and coefficients may be related to the test environments. When migrating from one region to another region, these parameters and coefficients may need to be adjusted slightly to obtain better recognition results. However, if the training data are sufficient, these parameters and coefficients will yield global values or local optimal values classified by region in the form of a list. The same situation applies to the weight file of the classifier. The proposed approach at this time can address the variability of the scenarios as much as possible and no additional setting is required.

In future work, the model migration research for embedded devices, such as Raspberry Pi with ARM chip, will be investigated, and a comparative analysis will be performed with the existing ADAS products.

5 CONCLUSION

We proposed an approach to detect traffic lights by using a combination of image processing and deep learning. First, we designed a detector to create candidate boxes that can cover all traffic lights based on image processing. In the process, the perspective relationship and the fractal dimension were both considered to dramatically reduce the number of invalid candidate boxes. Then, the candidate boxes were transformed into classified boxes by using the classifier. Finally, the traffic light boxes were produced from the classified boxes via postprocessing. Overall, the approach occupies a small amount of storage space.

We trained and validated the proposed approach on our dataset and the LaRA dataset. Several state-of-the-art methods were employed as the reference approaches. The results showed that the performance of the proposed approach in terms of recognizing traffic lights was satisfactory. In addition, estimation of the light boundary by the proposed approach was approbatory.

The red traffic light recognition performance achieved by the proposed approach was inferior for green traffic lights but superior for yellow traffic lights. The proposed approach has prominent advantages in recognizing small traffic lights.

Compared with the reference models, the proposed approach has a significant competitive advantage in terms of the recognition accuracy. Under the current serial program architecture, the proposed approach was relatively disadvantaged in processing video images. However, the runtime of the proposed approach can be further greatly reduced in future work by using parallel computing to carry out optimization.

Furthermore, the proposed approach was very small regarding the model size. The file size of the proposed approach including the CNN weight file was less than 10 MB. Thus, this approach can be used on smart terminals, mobile devices or embedded devices in the future.

Acknowledgement

This project was supported by the Scientific Research Foundation for the introduction of talents in Nantong University (No. 17R27), Youth Project of Natural Science Foundation of Jiangsu Province (No. BK20170440) and National Natural Science Foundation of China (No. 41501422). We thank to Feitao Ma, an engineer from Alibaba Group, for his help in adjusting the training samples.

REFERENCES

[1] Diaz, M.—Cerri, P.—Pirlo, G.—Ferrer, M. A.—Impedovo, D.: A Survey on Traffic Light Detection. In: Murino, V., Puppo, E., Sona, D., Cristani, M., Sansone, C. (Eds.): New Trends in Image Analysis and Processing – ICIAP 2015 Workshops (ICIAP 2015). Springer, Cham, Lecture Notes in Computer Science, Vol. 9281, 2015, pp. 201–208, doi: 10.1007/978-3-319-23222-5_25.

[2] Ivanchenko, V.—Coughlan, J.—Shen, H.: Real-Time Walk Light Detection with a Mobile Phone. In: Miesenberger, K., Klaus, J., Zagler, W., Karshmer, A. (Eds.): Computers Helping People with Special Needs (ICCHP 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6180, 2010, pp. 229–234, doi: 10.1007/978-3-642-14100-3_34.

[3] Vu, A.—Ramanandan, A.—Chen, A.—Farrell, J. A.—Barth, M.: Real-Time Computer Vision/DGPS-Aided Inertial Navigation System for Lane-Level Vehicle Navigation. IEEE Transactions on Intelligent Transportation Systems, Vol. 13, 2012, No. 2, pp. 899–913, doi: 10.1109/TITS.2012.2187641.

[4] Che, M.—Wang, Y.—Zhang, C.—Cao, X.: An Enhanced Hidden Markov Map Matching Model for Floating Car Data. Sensors (Basel), Vol. 18, 2018, Art. No. 1758, 19 pp., doi: 10.3390/s18061758.

[5] van de Sande, K. E. A.—Uijlings, J. R. R.—Gevers, T.—Smeulders, A. W. M.: Segmentation as Selective Search for Object Recognition. 2011 International Conference on Computer Vision (ICCV), 2011, pp. 1879–1886, doi: 10.1109/ICCV.2011.6126456.

[6] Uijlings, J. R. R.—van de Sande, K. E. A.—Gevers, T.—Smeulders, A. W. M.: Selective Search for Object Recognition. International Journal of Computer Vision, Vol. 104, 2013, No. 2, pp. 154–171, doi: 10.1007/s11263-013-0620-5.

[7] Zitnick, C. L.—Dollár, P.: Edge Boxes: Locating Object Proposals from Edges. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (Eds.): Computer Vision – ECCV 2014. Springer, Cham, Lecture Notes in Computer Science, Vol. 8693, 2014, pp. 391–405, doi: 10.1007/978-3-319-10602-1_26.

[8] Kim, H. K.—Yoo, K. Y.—Park, J. H.—Jung, H. Y.: Traffic Light Recognition Based on Binary Semantic Segmentation Network. Sensors (Basel), Vol. 19, 2019, No. 7, Art. No. 1700, 15 pp., doi: 10.3390/s19071700.

[9] de Charette, R.—Nashashibi, F.: Real Time Visual Traffic Lights Recognition Based on Spot Light Detection and Adaptive Traffic Lights Templates. IEEE Intelligent Vehicles Symposium, 2009, pp. 358–363, doi: 10.1109/IVS.2009.5164304.

[10] Diaz-Cabrera, M.—Cerri, P.—Medici, P.: Robust Real-Time Traffic Light Detection and Distance Estimation Using a Single Camera. Expert Systems with Applications, Vol. 42, 2015, No. 8, pp. 3911–3923, doi: 10.1016/j.eswa.2014.12.037.

[11] Haltakov, V.—Mayr, J.—Unger, C.—Ilic, S.: Semantic Segmentation Based Traffic Light Detection at Day and at Night. In: Gall, J., Gehler, P., Leibe, B. (Eds.): Pattern Recognition (DAGM 2015). Springer, Cham, Lecture Notes in Computer Science, Vol. 9358, 2015, pp. 446–457, doi: 10.1007/978-3-319-24947-6_37.

[12] Chen, Z.—Huang, X.: Accurate and Reliable Detection of Traffic Lights Using Multiclass Learning and Multiobject Tracking. IEEE Intelligent Transportation Systems Magazine, Vol. 8, 2016, No. 4, pp. 28–42, doi: 10.1109/MITS.2016.2605381.

[13] Islam, K. T.—Raj, R. G.: Real-Time (Vision-Based) Road Sign Recognition Using an Artificial Neural Network. Sensors (Basel), Vol. 17, 2017, No. 4, Art. No. 853, 32 pp., doi: 10.3390/s17040853.

[14] Zhou, X.—Yuan, J.—Liu, H.: Real-Time Traffic Light Recognition Based on C-HOG Features. Computing and Informatics, Vol. 36, 2017, No. 4, pp. 793–814, doi: 10.4149/cai_2017_4_793.

[15] John, V.—Yoneda, K.—Qi, B.—Liu, Z.—Mita, S.: Traffic Light Recognition in Varying Illumination Using Deep Learning and Saliency Map. 17th International IEEE Conference on Intelligent Transportation Systems (ITSC), 2014, pp. 2286–2291, doi: 10.1109/ITSC.2014.6958056.

[16] Ren, F.—Huang, J.—Jiang, R.—Klette, R.: General Traffic Sign Recognition by Feature Matching. 2009 24th International Conference Image and Vision Computing, New Zealand, 2009, pp. 409–414, doi: 10.1109/IVCNZ.2009.5378370.

[17] Kim, H. K.—Shin, Y. N.—kuk, S. G.—Park, J. H.—Jung, H. Y.: Night-Time Traffic Light Detection Based on SVM with Geometric Moment Features. International Journal of Computer and Information Engineering, Vol. 7, 2013, No. 4, pp. 472–475.

[18] Gómez, A. E.—Alencar, F. A. R.—Prado, P. V.—Osório, F. S.—Wolf, D. F.: Traffic Lights Detection and State Estimation Using Hidden Markov Models. IEEE Intelligent Vehicles Symposium Proceedings, 2014, pp. 750–755, doi: 10.1109/IVS.2014.6856486.

[19] Weber, M.—Wolf, P.—Zöllner, J. M.: DeepTLR: A Single Deep Convolutional Network for Detection and Classification of Traffic Lights. IEEE Intelligent Vehicles Symposium (IV), 2016, pp. 342–348, doi: 10.1109/ivs.2016.7535408.

[20] Behrendt, K.—Novak, L.—Botros, R.: A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification. IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 1370–1377, doi: 10.1109/ICRA.2017.7989163.

[21] Girshick, R.—Donahue, J.—Darrell, T.—Malik, J.: Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Vol. 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81.

[22] Girshick, R.: Fast R-CNN. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440–1448, doi: 10.1109/ICCV.2015.169.

[23] Ren, S.—He, K.—Girshick, R.—Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Advances in Neural Information Processing Systems 28 (NIPS 2015), 2015, pp. 91–99.

[24] He, K.—Gkioxari, G.—Dollár, P.—Girshick, R.: Mask R-CNN. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980–2988, doi: 10.1109/ICCV.2017.322.

[25] Li, X.—Ma, H.—Wang, X.—Zhang, X.: Traffic Light Recognition for Complex Scene with Fusion Detections. IEEE Transactions on Intelligent Transportation Systems, Vol. 19, 2018, No. 1, pp. 199–208, doi: 10.1109/TITS.2017.2749971.

[26] Redmon, J.—Divvala, S.—Girshick, R.—Farhadi, A.: You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788, doi: 10.1109/CVPR.2016.91.

[27] Liu, W.—Anguelov, D.—Erhan, D.—Szegedy, C.—Reed, S.—Fu, C. Y.—Berg, A. C.: SSD: Single Shot Multibox Detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (Eds.): Computer Vision – ECCV 2016. Springer, Cham, Lecture Notes in Computer Science, Vol. 9905, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.

[28] Shafiee, M. J.—Chywl, B.—Li, F.—Wong, A.: Fast YOLO: A Fast You Only Look Once System for Real-Time Embedded Object Detection in Video. arXiv preprint arXiv:1709.05943, 2017.

[29] Redmon, J.—Farhadi, A.: YOLO9000: Better, Faster, Stronger. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6517–6525, doi: 10.1109/CVPR.2017.690.

[30] Redmon, J.—Farhadi, A.: YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767, 2018.

[31] Iandola, F. N.—Han, S.—Moskewicz, M. W.—Ashraf, K.—Dally, W. J.—Keutzer, K.: SqueezeNet: AlexNet-Level Accuracy with 50× Fewer Parameters and < 0.5 MB Model Size. arXiv preprint arXiv:1602.07360, 2016.

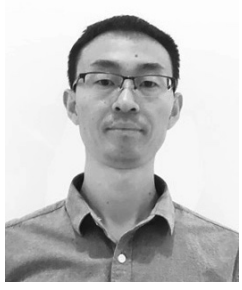
[32] Li, J.—Du, Q.—Sun. C.: An Improved Box-Counting Method for Image Fractal Dimension Estimation. Pattern Recognition, Vol. 42, 2009, No. 11, pp. 2460–2469, doi: 10.1016/j.patcog.2009.03.001.

[33] Rothe, R.—Guillaumin, M.—Van Gool, L.: Non-Maximum Suppression for Object Detection by Passing Messages Between Windows. In: Cremers, D., Reid, I., Saito, H., Yang, M. H. (Eds.): Computer Vision – ACCV 2014. Springer, Cham, Lecture Notes in Computer Science, Vol. 9003, 2014, pp. 290–306, doi: 10.1007/978-3-319-16865-4_19.

[34] Geirhos, R.—Rubisch, P.—Michaelis, C.—Bethge, M.—Wichmann, F. A.—Brendel, W.: ImageNet-Trained CNNs Are Biased Towards Texture; Increasing Shape Bias Improves Accuracy and Robustness. ICLR 2019, arXiv:1811.12231, 2019.

[35] Hosseinyalamdary, S.—Yilmaz, A.: A Bayesian Approach to Traffic Light Detection and Mapping. ISPRS Journal of Photogrammetry and Remote Sensing, Vol. 125, 2017, pp. 184–192, doi: 10.1016/j.isprsjprs.2017.01.008.

[36] Siogkas, G.—Skodras, E.—Dermatas, E.: Traffic Lights Detection in Adverse Conditions Using Color, Symmetry and Spatio-Temporal Information. International Conference on Computer Vision Theory and Applications – Volume 2: VISAPP (VISIGRAPP 2012), Rome, Italy, 2012, pp. 620–627, doi: 10.5220/0003855806200627.



Mingliang Che received his Ph.D. in cartography and geography information system from the University of Chinese Academy of Sciences in 2016. His current research interests include map matching, image processing and deep learning.

Mingjun Che received his B.Sc. Degree in computer science and technology from the Shandong Agricultural University in 2004. He currently works as an engineer in Wuxianshenghuo (Hangzhou) Info Tech Ltd. He does research in image processing, computer vision and artificial intelligence.

Zhenhua Chao is Associate Professor of the School of Geographic Sciences of Nantong University in China. He received his Ph.D. in remote sensing information extraction. His research interests are computer and information extraction. He has been involved in research areas in statistical downscaling and quantitative remote sensing for quite a time. He has published many papers in high quality publications.

Xinliang Cao received his M.Sc. Degree in software engineering from the University of Science and Technology of China in 2015. His current research interests include internet of things and artificial intelligence.

Computing and Informatics, Vol. 39, 2020, 464–480, doi: 10.31577/cai_2020_3_464

MEASURING SENTENCES SIMILARITY BASED ON DISCOURSE REPRESENTATION STRUCTURE

Mamdouh Farouk

Computer Science Department
Assiut University
Assiut, Egypt
e-mail: mamfarouk@aun.edu.eg

Abstract. The problem of measuring similarity between sentences is crucial for many applications in Natural Language Processing (NLP). Most of the proposed approaches depend on similarity of words in sentences. This research considers semantic relations between words in calculating sentence similarity. This paper uses Discourse Representation Structure (DRS) of natural language sentences to measure similarity. DRS captures the structure and semantic information of sentences. Moreover, the estimation of similarity between sentences depends on semantic coverage of relations of the first sentence in the other sentence. Experiments show that exploiting structural information achieves better results than traditional word-to-word approaches. Moreover, the proposed method outperforms similar approaches on a standard benchmark dataset.

Keywords: Sentence similarity, discourse representation structure, structural similarity

1 INTRODUCTION

Natural language processing has gained the focus of research especially after the explosion of data expressed in natural languages. Moreover, the wide use of social media and the need to analyze this data makes natural language tasks crucial. Measuring the similarity between natural language sentences is located at the core of many tasks to process natural language data. For instance, many approaches such as text classification, summarization, question answering, semantic search, and

plagiarism checking depend on sentence similarity [24, 33, 34]. Accuracy of calculation of sentences similarity affects these applications. Consequently, the problem of measuring the sentence similarity has got a lot of focus.

Measuring similarity between natural language sentences means estimating the degree of semantic relatedness between these sentences. The solutions for the problem of measuring sentence similarity still need improvement to accurately assess the similarity. Most of the previously proposed approaches depend on words of sentences. However, a sentence does not contain words only. Semantic relations between words are important components of a sentence.

Deep learning techniques that achieved good results in computer vision are also used in sentence similarity task. Word semantic representation is generated using deep learning techniques [20]. In this representation similar words have close vectors in the representation space. These numerical vector representations, normally of 300 length, for words, are used to get semantic similarity of sentences [21, 4].

Discourse Representation Theory (DRT) is a framework for representing the meaning of natural language sentences in a formal semantic approach [11]. DRT uses mental representation, which is DRS, to handle the meaning across sentence boundaries. DRT is used to implement language understanding systems [5]. DRS, which is used in DRT, consists of two main components: a set of discourse referents and a set of conditions. Consider this sentence "A woman walks. She smokes." This sentence can be represented in DRS as shown in Figure 1. The first line contains the set of referents (x and y). The other part is the set of conditions upon these referents.

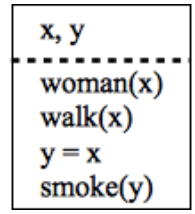


Figure 1. DRS representation for the sentence "A woman walks. She smokes."

This paper proposes a new approach for measuring sentence similarity. The proposed approach extracts semantic relations between words. Based on the similarity of semantic relations in sentences the similarity is calculated.

The main contribution of this work is calculating sentence structural similarity based on the semantic representation DRS that captures semantic and structural information of sentences. Unlike the traditional word-to-word approach, the proposed approach considers semantic relations between words in measuring similarity. Moreover, the proposed approach uses word embeddings to calculate the similarity between words.

The proposed approach is tested using standard datasets. Li2006 dataset [8] which is widely used in the evaluation of sentence similarity approaches is used to

evaluate the proposed approach. Moreover, MSRP dataset [3] which is used for paraphrase detection is also used to evaluate the performance of the proposed method. Experiments show that using DRS in sentence similarity improves measuring similarity.

The rest of this paper is organized as follows. Section 2 mentions the related work. Section 3 explains the proposed approach. A detailed example to get the similarity between two sentences is shown in Section 4. Section 5 describes the experiments and discusses the results. Finally, Section 5 concludes the presented work.

2 RELATED WORK

Different approaches have been proposed to calculate sentence similarity. Some of these approaches are string-based that consider the sentence as a sequence of characters. The similarity between two sequences of characters is assessed using string similarity methods such as q-grams [22] and Levenshtein distance [15].

Moreover, some approaches depend on word similarity to measure sentence similarity. These approaches consider the sentence as a set of words. WordNet [18], which is a lexical database for the English language, is widely used to find similarity between words. However, many approaches depend on analyzing big corpora to capture the semantics of words based on co-occurrences of words [24]. Latent Semantic Analysis (LSA) is one of the approaches that statistically analyzes big corpora to generate word semantic representation in a vector. Cosine similarity between these vectors measures the semantic similarity between words. Some approaches combine both methods (WordNet and corpus analyzing) to find similarity between sentences [25, 1].

The approaches for sentence similarity can be classified into three main classes: word-to-word based similarity, vector-based similarity, and structure-based approach [26]. In the word-to-word approach, the sentence similarity is calculated based on the similarity between the words in sentences. The second category depends on converting sentences to vectors that capture the semantic features of these sentences. Sentence similarity is calculated based on the similarity between these vectors. The third class of the approaches that measure sentence similarity is structure-based which exploits the structural information of sentences to calculate similarity.

Kenter and de Rijke in [21] propose an approach for measuring sentence similarity based on word embedding. They used word representation generated from deep learning to measure word similarity. Different pre-trained word vectors are used to measure sentence similarity. In addition to using word embeddings, TF-IDF weighting schema is used to consider word importance in the sentence. This approach is considered a word-to-word based approach. However, this approach ignores structural information of sentences. The structure of a sentence reveals important information that helps in the similarity measure.

Abdalgader and Skabar proposed to use word sense disambiguation and synonym expansion to improve sentence similarity [12]. Firstly, the sentences are processed and the meaning of the words are determined. A union vector for both sentences is constructed by finding the union of both sets of words. Additionally, the original set of words of each sentence is expanded by synonyms of the words belonging to this set. A vector representation for each sentence is constructed by finding the similarity between words in the union vector and words of that sentence vector. Finally, the cosine similarity between the two vectors is calculated as the semantic similarity between the two sentences. Although good results have been achieved using this approach, it needs external resources such as WordNet which is not available for all languages in high accuracy.

Some approaches combine different word similarity methods to calculate sentence similarity. For example, Li et al. in [25] proposed an approach to measure sentence similarity based on semantic net and corpus statistics. They computed semantic similarity between words based on a lexical database, which captures human knowledge, and based on a statistically analyzed corpus. In addition, word order similarity is calculated to measure order similarity for common words. A similar approach has been proposed by Pawar and Mago [1]. They combined WordNet and corpus analysis measures to assess sentence similarity. However, these approaches do not use structure information of sentences. In addition, external resources are needed to compute the similarity. The measured similarity depends on the accuracy of the used resources.

On the other hand, vector-based approaches generate vector representation for sentences and calculate similarity between these vectors. Skip-Thought [13] is a neural network model designed to train sentences and get vector representation that captures features of sentences. This model is similar to the skip-gram model that is used to get word vector representation. The idea is that similar sentences have similar features and close vectors. This model is used for sentence similarity systems. The input to their system is the words' vectors of sentences and the output is sentence vectors. These vectors are used to calculate sentence similarity.

Lee et al. in [14] introduced structure-based method to calculate sentence similarity. They extract grammar links from sentences and construct a grammar matrix in which rows represent links in the smaller (in length) sentence and columns represent links of the other sentence. Moreover, WordNet [18] is used to measure the similarity between words. The final similarity is calculated based on the constructed grammar matrix. Although this approach exploits lexical relations between words, it ignores semantic relations between words. Semantic relations are more helpful to assess semantic similarity between sentences.

Paraphrase detection is one task that is very related to sentence similarity. Recently Ferreira et al. proposed an approach for identifying paraphrases [27]. Their approach depends on extracting features and classifying a pair of sentences based on the extracted features. The extracted features are calculated based on lexical similarity, syntactic similarity, and semantic similarity. Lan and Xu proposed a learning-based approach that used sub-word level representation to detect para-

phrases [35]. However, these approaches can be used in paraphrase detection and do not assign a similarity value for a sentence pair. Moreover, these approaches do need labeled data.

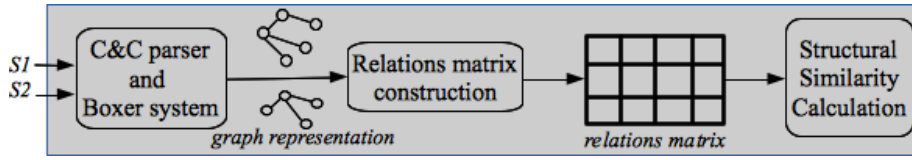


Figure 2. Proposed system architecture

3 SENTENCES SIMILARITY

A lot of NLP applications, such as social media analysis, question answering, and plagiarism, depend on sentence similarity. Consequently, the accuracy of measuring relatedness between sentences is a crucial task for many applications. The proposed approach exploits structure information in DRS representation of sentences to improve measuring sentence similarity. The input to the proposed system is two sentences and the output is a similarity value between 0 and 1.

As shown in Figure 2, calculating structural similarity between two sentences contains three steps. The first is generating DRS graphs for the inputted sentences. The second step is constructing a relation similarity matrix and the final step is calculating the structural similarity based on the relation matrix.

Structural information of a sentence helps to assess the sentence similarity [23]. Moreover combining structural similarity and word-based similarity improves the assessed similarity between sentences. As the first step for calculating structural similarity, each sentence is parsed and the output is passed to a semantic analyzer which outputs DRS graph representation equivalent to the sentence. Based on DRS graph representation of the sentences, a graph matching technique is used to measure the similarity between the two sentences. The following sub-sections explain the details of these steps.

3.1 Generation of Discourse Representation Structure

In order to get the structure of a sentence, a parser is used and semantic relations between words are extracted. A sentence semantic graph is constructed based on extracted relations. In this graph nodes represent words and edges represent semantic relations between words. The structural similarity of sentences is calculated based on the constructed graphs. In this paper, C & C parser [28] is used to parse sentences. In addition, the Boxer system [9] is used to get the semantic relations between sentence entities.

C & C parser contains many taggers such as Part Of Speech (POS) tagger and CCG supertagger. These taggers are highly efficient [9]. In addition, C & C contains Name Entity Recognizer which can determine ten different types of entities (organization, location, person, email, URL, first name, surname, title, quotation, and unknown name). C & C parser tags the words in a sentence with POS from the Penn treebank [17]. Then it builds sentence structure based on Combinatorial Categorial Grammar (CCG) paradigm. The output of the parsing is a syntax tree in which each node has POS tag, lemma, and name entity tag.

Based on the output of C & C parser, the Boxer system builds semantic representation for a sentence. The Boxer is a free software for analyzing text semantically. It depends on CCG and C & C parser to generate Discourse Representation Structure (DRS) for sentence text. DRS represents natural language text semantically. DRS captures the semantic of text and models it into related entities. DRS can be converted to other semantic formats such as first-order-logic [2]. The proposed approach uses semantic relations in DRS to calculate sentence similarity.

For example, consider these sentences: S_1 = "The boy who kills the snake is strong." and S_2 = "The boy is injured by a snake." The output of the Boxer system for these sentences is shown in Table 1. The DRS representation contains the words in sentences and the relations between words. For example the relation *theme* in S_1 connects the words *kills* and *snake*. Table 2 shows relations of both sentences.

Based on the output of the Boxer system, a semantic graph representation for the sentence is generated. Figure 3 shows the graph representation for the sentence S_1 = "The boy who kills a snake is strong." This graph captures the structure information of the sentence. Semantic relatedness between sentences is measured based on the generated graphs. Table 2 shows relations of DRS representation in both sentences.

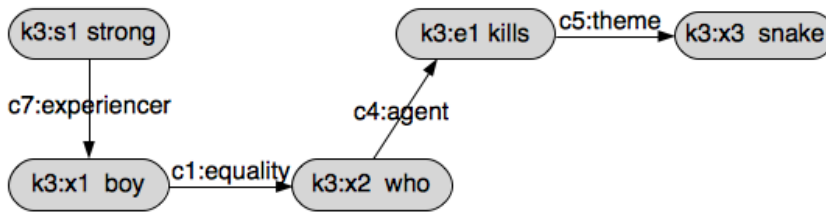


Figure 3. Sentence graph representation

3.2 DRS Graph Based Similarity

There are different techniques for solving graph matching problem. Graph matching is used in many applications in different fields [16]. For example, graph matching is used for measuring the similarity between documents [10]. In this paper structural sentence similarity is measured using sentences graphs. Based on the generated DRS

S_1 = "The boy who kills the snake is strong."	S_2 = "The boy is injured by a snake."
k3 attribute c6:strong:0 0 [ ]	k3 attribute c4:strong:0 0 [ ]
k3 concept c0:boy:0 0 [ ]	k3 concept c0:man:0 0 [ ]
k3 concept c2:snake:0 0 [ ]	k3 concept c5:snake:0 0 [ ]
k3 event c3:kill:0 0 [ ]	k3 event c1:kill:0 0 [ ]
k3 referent k3:e1 0 [ ]	k3 referent k3:e1 0 [ ]
k3 referent k3:s1 0 [ ]	k3 referent k3:s1 0 [ ]
k3 relation c1:equality 0 [ ]	k3 role c2:theme:1 0 [ ]
k3 role c5:theme:1 0 [ ]	k3 role c3:experiencer:-1 0 [ ]
k3 role c7:experiencer:1 0 [ ]	k3 role c6:agent:1 0 [ ]
k3 role c4:agent:-1 0 [ ]	k3 referent k3:x1 1 [The]
k3 referent k3:x1 1 [The]	c0:man:0 instance k3:x1 2 [man]
c0:boy:0 instance k3:x1 2 [boy]	k3 surface k3:e1 2 [is]
k3 referent k3:x2 1 [who]	k3:e1 main k3 1 [ ]
c3:kill:0 instance k3:e1 1 [kills]	c1:kill:0 instance k3:e1 3 [killed]
k3 referent k3:x3 1 [the]	k3 referent k3:x2 2 [a]
c2:snake:0 instance k3:x3 2 [snake]	c4:strong:0 arg k3:s1 1 [strong]
k3 surface k3:s1 2 [is]	c5:snake:0 instance k3:x2 4 [snake]
k3:s1 main k3 1 [ ]	c2:theme:1 int k3:e1 1 [ ]
c6:strong:0 arg k3:s1 3 [strong]	c2:theme:1 ext k3:x1 0 [ ]
c1:equality int k3:x1 3 [ ]	c3:experiencer:-1 int k3:x2 3 [ ]
c1:equality ext k3:x2 0 [ ]	c3:experiencer:-1 ext k3:s1 0 [ ]
c5:theme:1 int k3:e1 2 [ ]	c6:agent:1 int k3:e1 4 [ ]
c5:theme:1 ext k3:x3 0 [ ]	c6:agent:1 ext k3:x2 1 [by]
c7:experiencer:1 int k3:s1 1 [ ]	
c7:experiencer:1 ext k3:x1 0 [ ]	
c4:agent:-1 int k3:x2 2 [ ]	
c4:agent:-1 ext k3:e1 0 [ ]	

Table 1. DRS representation generated from Boxer system: (on left) DRS representation for sentence S_1 and (on right) DRS for sentence S_2

graphs for sentences, a relation matrix is constructed. Rows of this matrix represent relations of the first graph and columns represent relations of the second graph. Cell i, j in the matrix is filled with similarity value between relation i belonging to the first sentence and relation j belonging to the second sentence. Structural sentence similarity is calculated from this matrix.

3.2.1 Relation Similarity

As shown in Table 1, each relation has a name and links between the interior word and the exterior word. The similarity value between two relations is calculated in three steps:

Measuring the similarity between names of relations. The proposed approach distinguishes between the case when both relations have the same name

S_1 = "The boy who kills the snake is strong."	S_2 = "The boy is injured by a snake."
boy → **equality** → who	killed → **theme** → man
kills → **theme** → snake	snake → **experiencer** → strong
strong → **experiencer** → boy	killed → **agent** → snake
who → **agent** → kills	

Table 2. Relations of sentence S_1 and relations of sentence S_2 according to DRS representation

and the case when both relations have different names. The similarity value in the first case is higher than in the second case. If both relations have the same name the similarity value is 1. Otherwise the similarity value will be 0.7. This value has been assigned based on a tuning experiment using Li2006 dataset [8]. This value is working for every dataset.

Measuring similarity between interior nodes. Word embeddings [20] are used to calculate the similarity between interiors words of both relations. Word vectors which are trained on part of Google News[1] is used to get the word vector. The cosine similarity between words' vectors is calculated as the similarity between these words. In addition, word expansion is used to improve the word similarity measure. Two lists of words are obtained from the two words using expansion. The max similarity between these two lists is chosen as the similarity between the two words.

Measuring similarity between exterior nodes. Word embeddings are also used to find similarity between exterior words.

The following equation is used to calculate the similarity between two relations R_1 and R_2.

$$RelSim(R_1, R_2) = \frac{Sim(I_{R1}, I_{R2}) + Sim(E_{R1}, E_{R2})}{2} * NameSim(R_1, R_2). \quad (1)$$

$Sim(I_{R1}, I_{R2})$ is the similarity between interior word of R_1 and interior word of R_2. *NameSim* assesses similarity between names of relations.

For example, the similarity between *theme* relation in S_1 and *theme* relation in S_2 is calculated as follows:

- Similarity between names of relations is 1.
- Similarity between interior nodes: the interior word for *theme* relation in S_1 is *kills* and interior word for *theme* relation in S_2 is *killed*. $Sim(kills, Killed)$ is 0.94.
- Similarity between exterior nodes: $Sim(snake, man)$ is 0.08.

The final similarity between these relations is calculated according to Equation (1).

[1] This data set is publicly available at `https://code.google.com/archive/p/word2vec/`

3.2.2 Word Expansion

The proposed approach uses word expansion when measuring the similarity between two words. A word can be considered an expansion to another if there is an *equality* relation between them. For example, in Figure 3 the word *boy* can be used as expansion to the word *who*. When measuring the similarity between a word w_1 and another w_2, the proposed approach gets a list of words equal to w_1 and a list of words equal to w_2. The similarity between all words in the two lists is calculated and the max similarity is selected to represent the similarity between w_1 and w_2.

3.2.3 Calculating Structural Similarity

The proposed approach calculates the structural similarity by guessing to what extent the relations of sentence S_1 are covered by sentence S_2. This can be calculated based on the constructed matrix. In order to measure coverage of a relation belonging to the first sentence in the second sentence, the maximum similarity between this relation and all relations in the second sentence is selected. The structural similarity between S_1 and S_2 is calculated as follows:

$$Sim_{st}(S_1, S_2) = \frac{\sum_i^n maxSim(Ri, S2) * W_{Ri}}{\sum_i^n W_{Ri}} \tag{2}$$

where n is the number of relations in S_1 and W_{Ri} is the weight for the relation R_i. The weight of relations is used to reflect the importance of different relations according to its effect on the sentence meaning. Since the generated relations are limited, a fixed weight is assigned to each relation (Table 3). Common semantic roles have a high weight. Relations such as *agent* and *theme* have higher weights than other relations.

Relation Name	Weight
agent	8
theme	8
experiencer	6
is	4
in	3
other relations	1

Table 3. Weights for relations

4 EXPERIMENTS

The proposed approach has been implemented and tested against standard datasets to prove its effectiveness. The implemented system takes two sentences in natural language as input and measures the similarity between them. The output value

R & G Number	Human Assessment	Li 2006 [25]	Islam [7]	Pawar [1]	Omiotis [6]	Grammar Based [14]	Farouk [4]	Proposed Approach
1	0.01	0.33	0.06	0.023	0.11	0.22	0.104	0.121
5	0.005	0.29	0.11	0.07	0.10	0.06	0.12	0.161
9	0.005	0.21	0.07	0.015	0.10	0.35	0.087	0.067
13	0.108	0.53	0.16	0.292	0.30	0.32	0.204	0.207
17	0.048	0.36	0.26	0.366	0.30	0.41	0.246	0.317
21	0.043	0.51	0.16	0.231	0.24	0.44	0.276	0.178
25	0.065	0.55	0.33	0.279	0.30	0.07	0.30	0.271
29	0.013	0.34	0.12	0.133	0.11	0.20	0.243	0.188
33	0.145	0.59	0.29	0.762	0.49	0.07	0.244	0.423
37	0.13	0.44	0.2	0.10	0.11	0.07	0.218	0.276
41	0.28	0.43	0.09	0.045	0.11	0.02	0.264	0.203
47	0.35	0.72	0.3	0.161	0.22	0.25	0.332	0.283
48	0.355	0.64	0.34	0.54	0.53	0.79	0.386	0.317
49	0.29	0.74	0.15	0.299	0.57	0.38	0.397	0.288
50	0.47	0.69	0.49	0.253	0.55	0.07	0.175	0.378
51	0.14	0.65	0.28	0.302	0.52	0.39	0.133	0.303
52	0.485	0.49	0.32	0.842	0.6	0.84	0.428	0.387
53	0.483	0.39	0.44	0.89	0.5	0.18	0.382	0.433
54	0.36	0.52	0.41	0.783	0.43	0.32	0.286	0.24
55	0.405	0.55	0.19	0.315	0.43	0.38	0.243	0.402
56	0.59	0.76	0.47	0.977	0.93	0.62	0.489	0.521
57	0.63	0.7	0.26	0.477	0.61	0.82	0.318	0.359
58	0.59	0.75	0.51	0.892	0.74	0.94	0.388	0.496
59	0.86	1	0.94	0.856	1	1	0.889	0.80
60	0.58	0.66	0.60	0.898	0.93	0.89	0.549	0.484
61	0.52	0.66	0.29	0.934	0.35	0.08	0.265	0.339
62	0.77	0.73	0.51	1	0.73	0.94	0.594	0.46
63	0.56	0.64	0.52	0.7	0.79	0.95	0.367	0.525
64	0.955	1	0.93	0.873	0.93	1	0.876	0.85
65	0.65	0.83	0.65	0.854	0.82	–	0.578	0.597

Table 4. Results of the proposed approach and other approaches using Li2006 dataset

of the implemented system is between 0 to 1 (0 means no similarity and 1 means completely similar). In order to show the impact of using DRS of sentences, the proposed system is compared to other systems using standard datasets.

4.1 Li2006 Dataset

A short text semantic similarity benchmark dataset [8] is used to evaluate the proposed system. It is one of the widely used datasets in sentence similarity evaluation [25, 14, 7]. Originally, this dataset was created by Rubenstein and Goode-

Method	Pearson Correlation	Spearman Correlation
Li 2006	0.815	0.812
Islam	0.846	0.83
Pawar	0.781	0.823
Omiotis	0.857	0.889
Grammar based	0.714	0.639
word embedding	0.852	0.81
proposed approach	**0.872**	**0.894**

Table 5. Comparison between the proposed method and other methods

nough to measure word similarity [19]. The original dataset contains 65 pairs of words. Li et al. [25] added the definition of each word using the Collins Cobuild dictionary to use this dataset in sentence similarity. These 65 pairs of sentences are manually graded by 32 English native speakers according to the similarity degree.

The proposed system has been fed with pairs of sentences from Li2006 dataset. For each pair of sentences a similarity degree is returned. Table 4 shows the results of the proposed system along with other previously proposed systems. The results are shown in Table 4 for a subset of the selected benchmark. This subset contains 30 pairs of sentences selected carefully to cover different similarity ranges [14]. The proposed system is compared with classic approaches that do not use labeled data such as word-to-word or structure-based approaches. The results of Li approach [25], STS Meth [7] which integrates different word similarity methods, Pawar [1] which combines WordNet and corpus analysis to measure sentences similarity, Omiotis system [6] which is a new measure of semantic relatedness between texts, are included in Table 4. In addition, a similar approach to the proposed system which uses grammar-based similarity technique [14] is also included in the comparison. Moreover, results of Farouk's approach [4] which uses word embeddings in measuring the similarity are also included in Table 4. The Pearson correlation coefficient is calculated between each system results and human rating. Equation (3) is used to calculate the correlation between the human rating and the proposed system.

$$r = \frac{n\sum_i^n x_i y_i - \sum_i^n x_i \sum_i^n y_i}{\sqrt{n\sum_i^n x_i^2 - (\sum_i^n x_i)^2}\sqrt{n\sum_i^n y_i^2 - (\sum_i^n y_i)^2}} \tag{3}$$

where n is the number of sentence pairs, x is the similarity value of the proposed approach and y is human similarity value. The proposed approach has achieved the best results comparing to other systems in Table 4. The proposed system achieved 0.872 Pearson correlation with human similarity.

In addition, Spearman correlation is calculated to show the relationship between the results of different systems and human measured similarity. Equation (4)

explains how to calculate Spearman correlation.

$$r_s = 1 - \frac{6 \sum_i^n D_i}{n^3 - n^2} \tag{4}$$

where n is the number of samples and D is the difference between the human assessment and the system assessment. As shown in Table 5, the proposed system achieves the best results among all other systems.

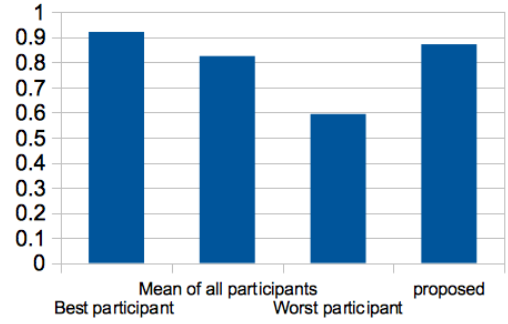


Figure 4. Results of the proposed system comparing to human raters

Figure 4 shows the achieved results comparing to human results in the Li2006 dataset. After calculating the average assessment of all human participants, Pearson correlation is calculated between the average assessment and each individual human assessment. As shown in Figure 4, the worst correlation between all participants is 0.594. The proposed approach achieves better than the mean of all human participants.

4.2 MSRP Dataset

Microsoft Research Paraphrase dataset [3] is widely used to evaluate sentence similarity techniques. It contains more than 5 000 pairs of sentences. It was partitioned into two sets. The first set contains around 4 200 pairs of sentences and is used as a training set. The other set contains around 1 700 pairs of sentences and is used for testing. Each pair is labeled by 1 (paraphrased) or 0 (not paraphrased). In our experiment the testing set is used to test the proposed approach.

In this experiment the proposed approach calculates the similarity between each pair of sentences and assigns a value between 0 and 1. A threshold value should be used to convert the calculated similarity value to 0 or 1. If the calculated similarity value is above the threshold, this pair is considered as the paraphrases. Different threshold values have been used previously in the literature. Omiotis approach used 0.2 as a threshold value [6], and 0.5 is used by Achananuparp in [32]. A tuning

experiment using hill-climbing algorithm [33] on MSRP training dataset determined that 0.45 is the threshold value for the proposed approach.

Metric	Accuracy	Precision	Recall	F-measure
Islam	72.64	74.65	89.13	81.25
Omiotis	69.97	70.78	93.40	80.52
grammar based	71.02	73.90	91.07	81.59
Farouk	71.6	76.2	83.3	79.6
proposed approach	70.46	72.34	89.30	79.93

Table 6. Results of the proposed approach and other approaches using MSRP dataset

Table 6 shows the achieved results and other approaches results in the MSRP dataset. Although the results of the proposed system are not the best, these results are comparable to other systems.

4.3 Results and Discussion

The experiments show that measuring similarity based on structural information of sentences gives better results than the traditional word-to-word approach. The proposed approach which uses C & C parser and the Boxer system to generate DRS representation for sentences outperforms other systems in Pearson correlation and spearman correlation measures.

However, the proposed approach achieves 70 % accuracy on the MSRP dataset. The results of MSRP dataset are not very good such as Li2006 dataset. This is because the proposed approach depends on the structure of sentences. The better structure of sentences the better performance of the proposed system. The first dataset (Li2006) is derived from a dictionary which means its sentences are well-structured. Consequently, the proposed approach achieves good results. However, the second dataset (MSRP) is derived from news sources on the web. This may explain the results of the second experiment.

Although the proposed approach outperforms other classic approaches in Li2006 dataset, it is sensitive to sentence structure. The performance of the proposed system will not be in the same high level with data that loose structure such as twitter messages. Moreover, DRS representation can be generated for many languages such as French [29], Chinese [30]. The proposed approach can be applied to other languages if DRS representation can be generated for that language.

5 CONCLUSION

The problem of finding the similarity between natural language sentences is important for many applications. Moreover, the structure of a sentence can reveal important information that helps in measuring sentence similarity. The proposed approach exploits structural information to calculate sentence similarity.

The proposed approach uses C & C parser and the Boxer system to generate DRS representation for sentences. This semantic representation captures the relations between words. Sentence similarity calculated based on the similarity between relations in both sentences. Moreover, word embedding is used to measure the similarity between words of relations. Experiments using standard datasets show the effectiveness of the proposed approach. The proposed approach performs well especially in case of well-structured sentences. Moreover, the proposed system achieves 0.872 Pearson correlation with human similarity. The proposed system outperforms other classical systems that depend on word-to-word and structure-based similarity.

REFERENCES

[1] Pawar, A.—Mago, V.: Calculating the Similarity Between Words and Sentences Using a Lexical Database and Corpus Statistics. 2018, arXiv preprint arXiv:1802.05667.

[2] Bos, J.: Wide-Coverage Semantic Analysis with Boxer. Proceedings of the 2008 Conference on Semantics in Text Processing (STEP '08), 2008, pp. 277–286, doi: 10.3115/1626481.1626503.

[3] Dolan, W. B.—Quirk, C.—Brockett, C.: Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004), Geneva, Switzerland, 2004, pp. 350–356.

[4] Farouk, M.: Sentence Semantic Similarity Based on Word Embedding and WordNet. 2018 13th International Conference on Computer Engineering and Systems (ICCES), 2018, pp. 33–37, doi: 10.1109/ICCES.2018.8639211.

[5] Guzman, F.—Joty, S.—Marquez, L.—Nakov, P.: Using Discourse Structure Improves Machine Translation Evaluation. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2014, pp. 687–698, doi: 10.3115/v1/P14-1065.

[6] Tsatsaronis, G.—Varlamis, I.—Vazirgiannis, M.: Text Relatedness Based on a Word Thesaurus. Journal of Artificial Intelligence Research, Vol. 37, 2010, No. 1, pp. 1–39, doi: 10.1613/jair.2880.

[7] Islam, A.—Inkpen, D.: Semantic Text Similarity Using Corpus-Based Word Similarity and String Similarity. ACM Transactions on Knowledge Discovery from Data, Vol. 2, 2008, No. 2, Art. No. 10, doi: 10.1145/1376815.1376819.

[8] O'Shea, J.—Bandar, Z.—Crockett, K.—McLean, D.: Pilot Short Text Semantic Similarity Benchmark Data Set: Full Listing and Description. Computing, 2008, available at: `https://semanticsimilarity.files.wordpress.com/2011/09/trmmucca20081_5.pdf`.

[9] Curran, J. R.—Clark, S.—Bos, J.: Linguistically Motivated Large-Scale NLP with C & C and Boxer. Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions, 2007, pp. 33–36.

[10] Hammouda, K. M.—Kamel, M. S.: Phrase-Based Document Similarity Based on an Index Graph Model. 2002 IEEE International Conference on Data Mining (ICDM '02), 2002, pp. 203–210, doi: 10.1109/ICDM.2002.1183904.

[11] Kamp, H.—Reyle, U.: From Discourse to Logic: An Introduction to Modeltheoretic Semantics, Formal Logic and Discourse Representation Theory. Springer Netherlands, Studies in Linguistics and Philosophy, Vol. 42, 1993, doi: 10.1007/978-94-017-1616-1.

[12] Abdalgader, K.—Skabar, A.: Short-Text Similarity Measurement Using Word Sense Disambiguation and Synonym Expansion. In: Li, J. (Ed.): AI 2010: Advances in Artificial Intelligence. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6464, 2010, pp. 435–444, doi: 10.1007/978-3-642-17432-2_44.

[13] Kiros, R.—Zhu, Y.—Salakhutdinov, R.—Zemel, R. S.—Torralba, A.—Urtasun, R.—Fidler, S.: Skip-Thought Vectors. In: Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., Garnett, R. (Eds.): Advances in Neural Information Processing Systems 28 (NIPS 2015), 2015, pp. 3294–3302.

[14] Lee, M. C.—Chang, J. W.—Hsieh, T. C.: A Grammar-Based Semantic Similarity Algorithm for Natural Language Sentences. The Scientific World Journal, Vol. 2014, 2014, Art. No. 437162, 17 pp., doi: 10.1155/2014/437162.

[15] Levenshtein, V. I.: Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. Soviet Physics – Doklady, Cybernetics and Control Theory, Vol. 10, 1966, No. 8, pp. 707–710.

[16] Farouk, M.—Ishizuka, M.—Bollegala, D.: Graph Matching Based Semantic Search Engine. In: Garoufallou, E., Sartori, F., Siatri, R., Zervas, M. (Eds.): Metadata and Semantics Research (MTSR 2018). Springer, Cham, Communications in Computer and Information Science, Vol. 846, 2018, pp. 89–100, doi: 10.1007/978-3-030-14401-2_8.

[17] Marcus, M. P.—Santorini, B.—Marcinkiewicz, M. A.: Building a Large Annotated Corpus of English: The Penn Treebank. Computational Linguistics, Vol. 19, 1993, No. 2, pp. 313–330.

[18] Miller, G. A.: WordNet: A Lexical Database for English. Communications of the ACM, Vol. 38, 1995, No. 11, pp. 39–41, doi: 10.1145/219717.219748.

[19] Rubenstein, H.—Goodenough, J. B.: Contextual Correlates of Synonymy. Communications of the ACM, Vol. 8, 1965, No. 10, pp. 627–633, doi: 10.1145/365628.365657.

[20] Mikolov, T.—Chen, K.—Corrado, G.—Dean, J.: Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), 2013, arXiv preprint arXiv:1301.3781.

[21] Kenter, T.—de Rijke, M.: Short Text Similarity with Word Embeddings. Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM '15), 2015, pp. 1411–1420, doi: 10.1145/2806416.2806475.

[22] Ukkonen, E.: Approximate String-Matching with Q-Grams and Maximal Matches. Theoretical Computer Science, Vol. 92, 1992, No. 1, pp. 191–211, doi: 10.1016/0304-3975(92)90143-4.

[23] Ma, W.—Suel, T.: Structural Sentence Similarity Estimation for Short Texts. Proceedings of the Twenty-Ninth International Florida Artificial Intelligence Research Society Conference (FLAIRS), 2016, pp. 232–237.

[24] Gomaa, W. H.—Fahmy, A. A.: A Survey of Text Similarity Approaches. International Journal of Computer Applications, Vol. 68, 2013, No. 13, pp. 13–18, doi: 10.5120/11638-7118.

[25] Li, Y.—McLean, D.—Bandar, Z. A.—O'Shea, J. D.—Crockett, K.: Sentence Similarity Based on Semantic Nets and Corpus Statistics. IEEE Transactions on Knowledge and Data Engineering, Vol. 18, 2006, No. 8, pp. 1138–1150, doi: 10.1109/TKDE.2006.130.

[26] Farouk, M.: Measuring Sentences Similarity: A Survey. Indian Journal of Science and Technology, Vol. 12, 2019, No. 25, pp. 1–11, doi: 10.17485/ijst/2019/v12i25/143977.

[27] Ferreira, R.—Cavalcanti, G. D. C.—Freitas, F.—Lins, R. D.—Simske, S. J.—Riss, M.: Combining Sentence Similarities Measures to Identify Paraphrases. Computer Speech and Language, Vol. 47, 2018, pp. 59–73, doi: 10.1016/j.csl.2017.07.002.

[28] Clark, S.—Curran, J. R.: Wide-Coverage Efficient Statistical Parsing with CCG and Log-Linear Models. Computational Linguistics, Vol. 33, 2007, No. 4, pp. 493–552, doi: 10.1162/coli.2007.33.4.493.

[29] Le, N. L.—Haralambous, Y.—Lenca, P.: Towards a DRS Parsing Framework for French. Advances in Natural Language Processing, Granada, Spain, 2019.

[30] Wang, Q.—Zhang, L.: Formal Semantics of Chinese Discourse Based on Compositional Discourse Representation Theory. In: Deng, H., Miao, D., Wang, F. L., Lei, J. (Eds.): Emerging Research in Artificial Intelligence and Computational Intelligence (AICI 2011). Springer, Berlin, Heidelberg, Communications in Computer and Information Science, Vol. 237, 2011, pp. 470–476, doi: 10.1007/978-3-642-24282-3_65.

[31] Achananuparp, P.—Hu, X.—Shen, X.: The Evaluation of Sentence Similarity Measures. In: Song, I. Y., Eder, J., Nguyen, T. M. (Eds.): Data Warehousing and Knowledge Discovery (DaWaK 2008). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5182, 2008, pp. 305–316, doi: 10.1007/978-3-540-85836-2_29.

[32] Russell, S. J.—Norvig, P.: Artificial Intelligence: A Modern Approach. 2nd Edition. Prentice, Upper Saddle River, New Jersey, 2003.

[33] Ramírez-Noriega, A.—Juárez-Ramírez, R.—Jiménez, S.—Inzunza, S.—Martínez-Ramírez, Y.: ASHuR: Evaluation of the Relation Summary-Content without Human Reference Using ROUGE. Computing and Informatics, Vol. 37, 2018, No. 2, pp. 509–532, doi: 10.4149/cai_2018_2_509.

[34] Zhang, L.—Hu, X.: Word Combination Kernel for Text Classification with Support Vector Machines. Computing and Informatics, Vol. 32, 2013, No. 4, pp. 877–896.

[35] Lan, W.—Xu, W.: Character-Based Neural Networks for Sentence Pair Modeling. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2018), Volume 2 (Short Papers), 2018, pp. 157–163, doi: 10.18653/v1/N18-2025.

Mamdouh Farouk is Assistant Professor in Assiut University, Egypt. He received his Ph.D. degree in artificial intelligence from the University of Tokyo, Japan in 2012. He received his B.Sc. and M.Sc. degrees in computer science from Cairo University in 2001. His research interests include data representation and Arabic language processing.

Computing and Informatics, Vol. 39, 2020, 481–509, doi: 10.31577/cai_2020_3_481

IMPROVED MULTI-POPULATION DIFFERENTIAL EVOLUTION FOR LARGE-SCALE GLOBAL OPTIMIZATION

Yongjie Ma*, Lin Zhu, Yulong Bai

College of Physics and Electronic Engineering
Northwest Normal University, Lanzhou, Gansu, China
e-mail: myjmyj@nwnu.edu.cn

Abstract. Differential evolution (DE) is an efficient population-based search algorithm with good robustness, however, it is challenged to deal with high-dimensional problems. In this paper, we propose an improved multi-population differential evolution with best-and-current mutation strategy (mDE-bcM). The population is divided into three subpopulations based on the fitness values, each of subpopulations uses different mutation strategy. After crossover, mutation and selection, all subpopulations are updated based on the new fitness values of their individuals. An improved mutation strategy is proposed, which uses a new approach to generate base vector that is composed of the best individual and current individual. The performance of mDE-bcM is evaluated on a set of 19 large-scale continuous optimization problems, a comparative study is carried out with other state-of-the-art optimization techniques. The results show that mDE-bcM has a competitive performance compared to the contestant algorithms and better efficiency for large-scale optimization problems.

Keywords: Differential evolution, large-scale global optimization, multiple populations, best-and-current mutation strategy, random migration strategy

Mathematics Subject Classification 2010: 11Y16

* Corresponding author

1 INTRODUCTION

Large-scale global optimization (LSGO), characterized by its high-dimensional decision variables and time-consuming objective functions, is a ubiquitous and spontaneous process that frequently appears in the real world problems. It has been increasingly considered as one of the most challenging issues in engineering optimization and the most active research fields. In recent years, the corresponding technologies that calculate the global optimal solution for LSGO have got some significant progresses in mechanical design, environment engineering, chemical engineering design, bioinformatics, image processing and other fields.

Differential evolution (DE), first proposed by Storn and Price [1, 2], is one of the most competitive Evolutionary Algorithms (EAs) currently in use [3, 4], and it has been applied to solve LSGO [5, 6]. Like the standard EAs, as a population-based, heuristic and stochastic search technique, DE first generates an initial population randomly in the feasible solution space as candidate solutions, and then generates a next generation of population through crossover, mutation, selection and other evolutionary operations, which move the population toward the global optimum. Different from the standard EAs, the mutation operation of DE is realized by the linear combination of the base vector and the difference vector of some individuals. Because it does not need derivative information of solving problems [7], and it has strong robustness in complex function optimization, DE receives wide attention and application, however, with the size and complexity of LSGO growing dramatically, the performance of DE algorithm will be terribly deteriorated. In other words, due to the curse of dimensionality, DE has encountered great difficulties in dealing with high-dimensional LSGO problems.

To improve performance of EAs for LSGO, numerous methods based on DE, including DE's variants and hybridization, were proposed. To solve slow convergence and stagnation caused by high-dimensional in LSGO, DE algorithm is improved from nine mechanisms, such as strategies and their adaptations, subpopulations, etc. Due to the difference of solving problems, no algorithm has been a winner on all LSGO benchmark functions so far [6].

In this paper, we propose a multi-populations DE with best-and-current mutation strategy, called mDE-bcM, aimed at improving the search precision and solving large-scale optimization problems. The mDE-bcM divides the population into three subpopulations based on the fitness value, each subpopulation uses a given mutation strategy. We propose a new improved mutation strategy that uses a linear combination of the best vector and current vector to produce the base vector. The mutation scale factor is not a fixed value, but a random one to improve the diversity of the population. After one evolution, every subpopulation updates on the base of the new fitness value rank. And this technology mitigates the risk of lower diversity, enhances the rate of convergence, promotes the whole algorithm performance. This paper also uses two different strategies on three subpopulations, one is elitism and the other is random migration. Elitism strategy keeps one subpopulation which has good fitness values from participating in the evolution of the next generation, and

it maintains the other two subpopulations to update. Random migration strategy randomly selects ten individuals in one subpopulation which has good fitness values, five of them exchange with a random sample of five individuals in the general population, another five exchange with a random sample of five individuals in the poor population.

The mDE-bcM performance is evaluated on two sets of large-scale global optimization benchmark functions with dimensions ranging from 50 to 1 000, the results show that mDE-bcM is capable to solve continuous large-scale problems effectively and produce competitive results when compared with the state-of-the-art algorithms which are designed specifically to solve such problems.

The rest of the paper is structured as follows. Classic DE is introduced in Section 2, a brief review of related works on multi-populations and mutation strategy is presented in Section 3, the proposed algorithm (mDE-bcM) is introduced in Section 4, experimental set-up is presented in Section 5, experimental results and analysis are presented in Section 6. Finally, the work is concluded in Section 7.

2 CLASSIC DIFFERENTIAL EVOLUTION

2.1 Initialization

An initial population $X_0 = \{\mathbf{x}_1, \ldots, \mathbf{x}_{NP}\}$ in DE is randomly generated in the entire search space (unconstrained) or the feasible solution space (constrained). Based on real number coding, the initial value of the j^{th} decision variable of the i^{th} individual at generation $g = 0$ is generated among the prescribed lower bound $\mathbf{x}_{min} = \{x^j_{min} \mid j = 1, \ldots, D\}$ and upper bound $\mathbf{x}_{\max} = \{x^j_{\max} \mid j = 1, \ldots, D\}$ by:

$$x^j_{i,0} = x^j_{\min} + \text{rand}(0,1) \cdot (x^j_{\max} - x^j_{\min}), \quad i = 1, 2, \ldots, NP, \quad j = 1, 2, \ldots, D \quad (1)$$

where D is the dimensions of the problem, NP is population size, $\text{rand}(0,1)$ is a uniformly distributed random variable within the range $(0,1)$.

2.2 Mutation Operation

In DE, the variation vector $\mathbf{v}_{i,g} = \{v^j_{i,g} \mid j = 1, \ldots, D\}$ is composed of a base vector $\mathbf{x}_{i,r1,g} = \{x^j_{i,r1,g} \mid j = 1, \ldots, D\}$ and the differential vector $\mathbf{x}_{i,r2,g} - \mathbf{x}_{i,r3,g}$ of two random individuals $\mathbf{x}_{i,r2,g}$ and $\mathbf{x}_{i,r3,g}$ in the g^{th} generation population, the differential vector $\mathbf{x}_{i,r2,g} - \mathbf{x}_{i,r3,g}$ is scaled by the scale factor F. How to generate the variation vector is called mutation strategy, the most commonly used mutation strategies are summarized in [8]:

- DE/rand/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,r1,g} + F \cdot (\mathbf{x}_{i,r2,g} - \mathbf{x}_{i,r3,g}) \quad (2)$$

- DE/rand/2:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,r1,g} + F \cdot (\mathbf{x}_{i,r2,g} - \mathbf{x}_{i,r3,g}) + F \cdot (\mathbf{x}_{i,r4,g} - \mathbf{x}_{i,r5,g}) \quad (3)$$

- DE/best/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{\text{best},g} + F \cdot (\mathbf{x}_{i,r1,g} - \mathbf{x}_{i,r2,g}) \tag{4}$$

- DE/best/2:

$$\mathbf{v}_{i,g} = \mathbf{x}_{\text{best},g} + F \cdot (\mathbf{x}_{i,r1,g} - \mathbf{x}_{i,r2,g}) + F \cdot (\mathbf{x}_{i,r3,g} - \mathbf{x}_{i,r4,g}) \tag{5}$$

- DE/rand-to-best/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + K \cdot (\mathbf{x}_{\text{best},g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{i,r1,g} - \mathbf{x}_{i,r2,g}) \tag{6}$$

- DE/rand-to-best/2:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + K \cdot (\mathbf{x}_{\text{best},g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{i,r1,g} - \mathbf{x}_{i,r2,g} + \mathbf{x}_{i,r3,g} - \mathbf{x}_{i,r4,g}) \tag{7}$$

- DE/current-to-rand/1:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + K \cdot (\mathbf{x}_{i,r1,g} - \mathbf{x}_{i,g}) + F \cdot (\mathbf{x}_{i,r2,g} - \mathbf{x}_{i,r3,g}). \tag{8}$$

The parameters $r1$, $r2$, $r3$, $r4$ and $r5$ are exclusive integers generated randomly within the range of $[1, NP]$, $\mathbf{x}_{\text{best},g}$ is the individual which has the best fitness value in the g^{th} generation population, $\mathbf{x}_{i,r1,g}$, $\mathbf{x}_{i,r2,g}$, $\mathbf{x}_{i,r3,g}$, $\mathbf{x}_{i,r4,g}$ and $\mathbf{x}_{i,r5,g}$ are target vectors selected from the g^{th} generation population, K is randomly selected in $[0, 1]$, F is the scale factor.

2.3 Crossover Operation

Trial vector $\mathbf{u}_{i,g} = \{u_{i,g}^j \mid j = 1, \ldots, D\}$ is generated by crossover operator. The DE generally employs two kinds of crossover methods, namely binomial crossover (bin) and exponential crossover (exp).

In binomial crossover, at least one component $u_{i,g}^j$ of trial vector $\mathbf{u}_{i,g}$ is provided randomly by the variation vector $\mathbf{v}_{i,g}$, that is as follows:

$$u_{i,g}^j = \begin{cases} v_{i,g}^j, & \text{rand}(0,1) \le CR \text{ or } j = j_{\text{rand}}, \\ x_{i,g}^j, & \text{otherwise}, \end{cases} \tag{9}$$

where $j = 1, \ldots, D$, j_{rand} is a randomly chosen integer in the range $[1, D]$, $CR \in (0, 1)$ is the crossover rate.

Exponential crossover operation is as follows:

$$u_{i,g}^j = \begin{cases} v_{i,g}^j, & j = \langle l \rangle_D, \langle l+1 \rangle_D, \ldots, \langle l+L-1 \rangle_D, \\ x_{i,g}^j, & \text{otherwise}, \end{cases} \tag{10}$$

where integer $l \in [1, D]$, integer $L \in [l, D]$ depends on the crossover probability (CR), the angular bracket $\langle l \rangle_D$ denotes a modulo function with modulus D. That

is to say, starting with l^{th} component, L components are continuously selected from the variation vector $\mathbf{v}_{i,g}$ as the component of trial vector $\mathbf{u}_{i,g}$, the remaining D-L components of trial vector $\mathbf{u}_{i,g}$ are from $\mathbf{x}_{i,g}$.

2.4 Selection Operation

The selection operation chooses the best individual according to the fitness value of target vector and trial vector in the g^{th} generation population, as shown in the following:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g}, & f(\mathbf{u}_{i,g}) < f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g}, & \text{otherwise}, \end{cases} \tag{11}$$

where $\mathbf{x}_{i,g+1}$ is the next generation target vector.

3 RELATED WORK

The DE is a simple, effective, population-based optimization technique, but it is inefficient for solving LSGO because of the computational complexity of high-dimensional LSGO [9]. Many researches have been devoted to solve these problems, and their research directions are mainly focused on multiple populations, mutation strategy, parameters control, population size adaptation, problem decomposition and so on, all of which are aimed at reducing the computational complexity in high-dimensional LSGO.

Multiple populations are one of the most widely used methods of problem decomposition, which can enhance the DE performance by parallel computing [10] and enhance the population diversity by dividing the population into independent subpopulations [11], meanwhile smaller subpopulations are efficient at quickly improving their fitness values, but smaller subpopulations might likely dissipate diversity and cause premature convergence [10].

Lampinen first tried to apply multiple populations to DE algorithm [12], who divided a population into multiple subgroups in DE calculation. Zaharie [13] proposed a DE with a multi-population approach. Weber et al. [14] proposed a parallel differential evolution, named SOUPDE, which was a multi-population algorithm with a differential evolution logic. In SOUPDE, the subpopulations quickly exploited some areas of the decision space, a new search logics, integrated shuffling and updating mechanisms were introduced into the subpopulation to avoid a diversity loss and premature convergence. Chen et al. [15] proposed a parallel differential evolution with multi-population and multi-strategy, which divided an initial population into three subpopulations with different mutation strategies, and the subpopulations evolved independently at first, and then communicated with each other at regular intervals. Ali et al. [11] proposed a multi-population DE algorithm, called mDE-bES, the population was divided into four independent subgroups, each with different mutation and update strategies, a novel mutation strategy that used information from either the best individual or a random individual was used, individuals be-

tween the subgroups were exchanged at the end of each function evaluations epoch. Wu et al. [16] proposed a multi-population ensemble DE, called MPEDE, which simultaneously consisted of three mutation strategies, i.e., "current-to-$p$best/1", "current-to-rand/1" and "rand/1". MPEDE had three equally sized smaller indicator subpopulations and one much larger reward subpopulation, each constituent mutation strategy has one indicator subpopulation. After some iterations, the best performing mutation strategy was determined according to the ratios between fitness improvements and consumed function evaluations, the reward subpopulation was allocated to the determined best performing mutation strategy dynamically.

All these studies show that population decomposition may be an effective and feasible method, which can not only reduce the complexity of calculation by reducing the population size, but also facilitate parallel calculation to improve the efficiency of calculation.

Mutation strategy also has a significant impact on the performance of DE, for each individual, the number of successful generations related to a certain mutation strategy [9]. Mutation strategy is represented by DE/x/y/z. Among them, x means the way of selecting vectors in mutation operation, its possible values are "Rand", "best", "current", "Rand to best", etc.; y represents the number of differential vectors in mutation operation, usually y = 1 or 2; z refers to crossover method, mainly including binomial (bin) and exponential (exp).

In order to improve the performance of DE algorithm, some researchers have proposed many improved mutation strategies in recent years.

Price et al. have proposed 10 mutation strategies of DE [2], including DE/rand /1/bin, DE/rand/1/exp, DE/best/1/bin, DE/best/1/exp, DE/rand/2/bin, DE /rand/2/exp, DE/best/2/bin, DE/best/2/exp, DE/rand-to-best/1/bin, DE /rand-to-best/1/exp, some of them have been described in Section 2.2.

Zhang et al. [17] proposed a new DE algorithm, called JADE, with a new mutation strategy "DE/current-to-$p$best/1" as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i \cdot (\mathbf{x}^p_{\text{best},g} - \mathbf{x}_{i,g}) + F_i \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{12}$$

where $\mathbf{x}^p_{\text{best},g}$ is an individual randomly chosen from the top $100p\,\%$ individuals in the current population, $p \in (0, 1]$, and F_i is the mutation factor randomly generated for each $\mathbf{x}_i$. The parameter adaptation in JADE automatically updated the control parameters, the "DE/current-to-$p$best/1" used the optional archive to provide information of progress direction by historical data.

Kong et al. [18] proposed mutation operator based on symbolic function strategy:

$$\mathbf{v}_{i,g} = (1 - w)\mathbf{x}_{i,g} + w \cdot (s_1\mathbf{x}_{\text{best},g} + s_2\mathbf{x}_{r1,g}) + F \cdot (\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}) \tag{13}$$

where w is a random number between 0 and 1, s_1 and s_2 are symbolic functions.

Zhang et al. [19] adopted a directional mutation operator, which could be combined with any other DE mutation strategy:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F \cdot \delta_{r2,g} \tag{14}$$

where $\delta_{r2,g}$ is a difference vector from the pool of difference vectors. If $\mathbf{u}_{i,g}$ will survive and become $\mathbf{x}_{i,g+1}$ after crossover operation, which shows that $\mathbf{u}_{i,g}$ must contain some better components than $\mathbf{x}_{i,g}$, the difference vector $\mathbf{u}_{i,g} - \mathbf{x}_{i,g}$ should be a descent direction at $\mathbf{x}_{i,g}$, and was saved into the pool of difference vectors.

Qiu et al. [20] proposed fractal mutation factor differential evolution (FMDE) algorithm, which consisted of an additional mutation factor simulated by a different Hurst index Fractal Brownian Motion (FBM), the proposed mutation strategy, improved from JADE's "DE/current-to-$p$best/1", was divided into two parts to reflect the changes of overall and random of the target population, it is defined as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + (F_i + F_{i,FBM}) \cdot (\mathbf{x}^p_{\text{best},g} - \mathbf{x}_{i,g} + \lambda \cdot (\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g})) \tag{15}$$

where F_i is variation factor of the overall changes, it changes near its position parameter μ_F and obeys cauchy distribution, $F_{i,FBM}$ is variation factor of the random changes, λ is an additional control parameter.

Raghav et al. [21] proposed a new "Memory based DE" (MBDE) where had two "swarm operators", and their operators based on the pBEST and gBEST mechanism of particle swarm optimization, their "Swarm Mutation" is as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + \left| \frac{f(p^{\text{BEST}}_{i,g})}{f(\mathbf{x}^{WORST}_{i,g})} \right| \left(p^{\text{BEST}}_{i,g} - \mathbf{x}_{i,g}\right) + \left| \frac{f(g^{\text{BEST}}_{g})}{f(\mathbf{x}^{WORST}_{i,g})} \right| \left(g^{\text{BEST}}_{g} - \mathbf{x}_{i,g}\right) \tag{16}$$

where $p^{\text{BEST}}_{i,g}$ is the personal best position of the vector $\mathbf{x}_{i,g}$, $g^{\text{BEST}}_{i,g}$ is the global best position of the vector $\mathbf{x}_{i,g}$, $f(p^{\text{BEST}}_{i,g})$, $f(g^{\text{BEST}}_{g})$ and $f(\mathbf{x}^{WORST}_{i,g})$ are the personal best function value, the global best function value and the worst function value of vector $\mathbf{x}_{i,g}$ in the current generation g, respectively.

Different mutation strategies adapt to different types of optimization problems. For example, DE/rand/1/exp is suitable for the optimization of multimodal functions, DE/best/1/exp is suitable for the optimization of unimodal functions. To further improve the performance of DE, multiple mutation strategies are used.

Elsayed et al. [22] proposed 4 new mutation strategies "DE/rand/3", "DE/best /3", "DE/rand-to-current/2" and "DE/rand-to-best and current/2" as follows:

- DE/rand/3:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F_i \cdot ((\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}) + (\mathbf{x}_{r4,g} - \mathbf{x}_{r5,g}) + (\mathbf{x}_{r6,g} - \mathbf{x}_{r7,g})) \tag{17}$$

- DE/best/3:

$$\mathbf{v}_{i,g} = \mathbf{x}_{\text{best},g} + F_i \cdot ((\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) + (\mathbf{x}_{r3,g} - \mathbf{x}_{r4,g}) + (\mathbf{x}_{r5,g} - \mathbf{x}_{r6,g})) \tag{18}$$

- DE/rand-to-current/2:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F_i \cdot ((\mathbf{x}_{r2,g} - \mathbf{x}_{i,g}) + (\mathbf{x}_{r3,g} - \mathbf{x}_{r4,g})) \tag{19}$$

- DE/rand-to-best and current/2:

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F_i \cdot ((\mathbf{x}_{\text{best},g} - \mathbf{x}_{r2,g}) + (\mathbf{x}_{r3,g} - \mathbf{x}_{r4,g})). \tag{20}$$

These mutation strategies used three or two differential vectors, which get the benefit of the arithmetic recombination and the random effect to investigate the search space for both separable and non-separable unimodal test problems.

Elsayed et al. used four mutation strategies on four subpopulations, respectively.

Tong et al. [23] proposed an improved multi-population ensemble DE (IMPEDE) and used a new mutation strategy "DE/$p$bad-to-$p$best/1" instead of the mutation strategy "DE/rand/1", "DE/$p$bad-to-$p$best/1" is defined as follows:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{pbest,g} - \mathbf{x}_{pbad,g}) \tag{21}$$

where $\mathbf{x}_{pbest,g}$, $\mathbf{x}_{pbad,g}$ are individuals randomly chosen from the best top or the bad top $100p\,\%$ individuals in the current population, respectively, $p \in (0, 1]$, the new strategy "DE/$p$bad-to-$p$best/1" utilized the information of the bad solution ($p$bad) and the good solution ($p$best) to balance exploration and exploitation.

Except "DE/$p$bad-to-$p$best/1" over subpopulation pop_1, Tong et al. used "DE /current-to-rand/1" over subpopulation pop_2 , "DE/$p$bad-to-$p$best/1" over subpopulation pop_3.

Xu et al. [24] proposed a new adaptive differential evolution named CAMDE, which defined two new multi-population-based mutation operators, denoted as "DE /current-to-nbest/u" and "DE/current-to-rand/u" as follows:

- DE/current-to-nbest/u:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{nbest,g} - \mathbf{x}_{i,g}) + F \cdot (\widetilde{\mathbf{x}}_{r1,g} - \mathbf{x}_{r2,g}) \tag{22}$$

- DE/current-to-rand/u:

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F \cdot (\mathbf{x}_{r2,g} - \mathbf{x}_{i,g}) + F \cdot (\widetilde{\mathbf{x}}_{r1,g} - \mathbf{x}_{r3,g}) \tag{23}$$

where $\mathbf{x}_{nbest,g}$ means the non-dominated best solution, and $\widetilde{\mathbf{x}}_{r1,g}$ is randomly chosen from the union $X_g \cup Y_g$ of the main population X_g and the external population Y_g, $Y_g = \{\mathbf{y}_{1,g}, \ldots, \mathbf{y}_{K,g}\}$ is composed of individuals better than the corresponding target vector $\mathbf{x}_{i,g}$, K is the maximal size of the population Y_g.

In CAMDE, each of two mutation operators is selected with equal probability.

4 IMPROVED MULTI-POPULATION DIFFERENTIAL EVOLUTION WITH BEST-AND-CURRENT MUTATION STRATEGY (MDE-BCM)

In DE algorithm, the reasonable control parameter setting and excellent mutation strategy directly affect the convergence of algorithm, the performance and reliability of the solution. Meanwhile, different strategies and parameters setting are

adapted to the different optimization problems. Optimizing different benchmarks with different characteristics such as uni-modal, multi-modal, continuous, discrete, low-dimension, high-dimension requires different mutation strategies and suitable parameters. For these reasons, the proposed algorithm in this paper divides the population into three subpopulations S1, S2 and S3 based on the fitness value, each of which follows a different mutation strategy and corresponding parameters. This technology could reduce the burden of parameter selection, increase the diversity of population, and improve the rate of convergence of the algorithm.

The structure of mDE-bcM algorithm is written in Algorithm 1.

Algorithm 1 Pseudo code of mDE-bcM

Set $CR = 0.9$, $F = 0.5$; Initialize NP, G_m, D; Set $g = 0$;
Initialize the population randomly distributed in the solution space;
for $i = 1$ to NP **do**
  Calculate $f(\mathbf{x}_{i,0})$ for each individual $\mathbf{x}_{i,0}$ in initial population X_0;
end for
Rank $\mathbf{x}_{i,0}$ based on $f(\mathbf{x}_{i,0})$;
Partition initial population X_0 into S1, S2, S3 with the same size $NP/3$;
while $(g < G_m)$ **do**
  for each $\mathbf{x}_{i,g}$ in Sk $(k = 1, 2, 3)$ **do**
    $\mathbf{v}_{i,g} = \text{mutation}(\mathbf{x}_{i,g})$;
    $\mathbf{u}_{i,g} = \text{crossover}(\mathbf{x}_{i,g}, \mathbf{v}_{i,g})$;
    $\mathbf{x}_{i,g+1} = \text{selection}(\mathbf{x}_{i,g}, \mathbf{u}_{i,g})$;
    Calculate $f(\mathbf{x}_{i,g+1})$ for each new individual $\mathbf{x}_{i,g+1}$ in subpopulation Sk;
  end for
  $X_{g+1} = \bigcup_1^k \text{S}k, (k = 1, 2, 3)$;
  Rank $\mathbf{x}_{i,g+1}$ based on $f(\mathbf{x}_{i,g+1})$;
  Partition X_{g+1} into new S1, S2, S3 with the same size;
  Update the best archive of S2 and S3 by selecting randomly 10 individuals from S1;
  Set $g = g + 1$;
end while

The mDE-bcM algorithm is described as follows.

4.1 Multiple Subpopulations

The mechanism of multi-population ensures that each subpopulation is not affected by the interference of other subpopulations in the process of evolution. Moreover, it also improves the diversity of the population to a certain degree. If the diversity of one subpopulation get worse, individuals with a large difference in the other two subpopulations do not get worse because they are evolved independently, the diversity of the population will be improved through migration among subpopula-

tions. Meanwhile, multi-population also makes it possible to parallelize which can effectively reduce the computational time.

4.2 Evolutionary Process

The mDE-bcM firstly initializes the entire population by randomly generating individuals with D dimensions, then the fitness values are calculated and sorted for all individuals. Based on the fitness values, the population is divided into three subpopulations that are S1, S2 and S3. the size of each subpopulation is $NP1 = NP/3$. S1 is a subpopulation with better fitness, S2 is a subpopulation with general fitness, S3 is a subpopulation with poor fitness. Three subpopulations evolve respectively and concurrently within each subpopulation. After crossover, mutation and selection, three subpopulations S1, S2 and S3 are combined into one population, the fitness values of individuals in the combined population are recalculated and all individuals are sorted, then combined population is divided into three subpopulations according to the fitness values. Finally, the algorithm enters the next iteration.

4.3 Best and Current Mutation Strategy

Traditional mutation strategy in dealing with high-dimensional problems can not get good convergence effect. Inspired by a greedy mutation strategy "DE/current-to-*p*best/1" in [11], we proposed a new mutation strategy called "DE/best-and-current/1".

Different from the classic mutation strategy which selects one of the population members as the base vector, and also different from mDE-bES in [11] which selects the best or a random individual as the base vector, we use a linear combination of the best vector $\mathbf{x}_{\text{best},g}$ and current vector $\mathbf{x}_{i,g}$. Mutation scale factor F also is a random value, instead of a fixed value. Mutant vector $\mathbf{v}_{i,g}$ is created as follows:

$$\mathbf{v}_{i,g} = (a_1\mathbf{x}_{\text{best},g} + a_2\mathbf{x}_{i,g}) + \text{rand}(0,1) \cdot (\mathbf{x}_{r_1,g} + \mathbf{x}_{r_2,g}) \tag{24}$$

where r_1, r_2 are random exclusive integers within the interval $[1, NP/3]$, a_1, a_2 are scalars randomly selected between $(0, 1)$, and $a_1 + a_2 = 1$.

In order to avoid the degradation of the offspring, S1 is an elitist-population, it adopts the traditional mutation strategy "DE/rand/1". S2 and S3 use a random migration strategy and user-defined best-and-current mutation strategy (as shown in formula (24), the random migration strategy randomly selects 10 individuals from S1, 5 of them are $\mathbf{x}_{\text{best},g}$ in S2 and 5 of them are $\mathbf{x}_{\text{best},g}$ in S3. Then, S1, S2 and S3 evolve in parallel using two different evolution strategies.

5 EXPERIMENTAL SET-UP

5.1 Benchmark Functions

The algorithm was tested on 19 benchmark large-scale global continuous optimization functions (F_1–F_{19}). The functions were taken from the special issue of Soft Computing on scalability of evolutionary algorithms [25, 26], the dimensions of these functions in the special issue were 50, 100, 200, 500 and 1 000, respectively, the optimal solutions for all these functions are known. Each function runs 25 times independently to evaluate the performance of the algorithm. The definitions of the functions F_1–F_{11} are shown in Table 1. In Table 2, the functions F_{12}–F_{19} are generated by hybridizing a non-separable function F_{ns} with other function F' using a splitting mechanism that defines the ratio of variables which are evaluated by F_{ns} using parameter m_{ns}.

F	Name	Definition	Range	$f(x^*)$
F_1	Shif. Sphere Function	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	-450
F_2	Shif. Schwefel Problem 2.21	$f_2(x) = \max_{i\in[1,D]} \lvert x_i\rvert$	$[-100, 100]^D$	-450
F_3	Shif. Rosenbrock's Function	$f_3(x) = \sum_{i=1}^{D}(100(x_i^2 + x_{i+1})^2 + (x_i - 1)^2)$	$[-100, 100]^D$	390
F_4	Shif. Rastrigin's Function	$f_4(x) = \sum_{i=1}^{D}[x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5, 5]^D$	-330
F_5	Shif. Griewank's Function	$f_5(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\frac{x_i}{\sqrt{i}} + 1$	$[-600, 600]^D$	-180
F_6	Shif. Ackley's Function	$f_6(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right)$ $-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^D$	-140
F_7	Shif. Schwefel's Problem 2.22	$f_7(x) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	$[-10, 10]^D$	0
F_8	Shif. Schwefel's Problem 1.2	$f_8(x) = \sum_{i=1}^{D}(\sum_{j=1}^{i} x_j)^2$	$[-65.536, 65.536]^D$	0
F_9	Shif. Extended f10	$f_9(x) = \left(\sum_{i=1}^{(D-1)} f_{10}(x_i, x_{i+1})\right) + f_{10}(x_D, x_1)$ $f_{10}(x) = (x^2 + y^2)^{0.25}(\sin^2(50(x^2 + y^2)^{0.1}) + 1)$	$[-100, 100]^D$	0
F_{10}	Shif. Bohachevsky	$f_{10}(x) = \sum_{i=1}^{D}[x_i^2 + 2x_{i+1}^2 - 0.3\cos(3\pi x_i)$ $-0.4\cos(4\pi x_{i+1}) + 0.7]$	$[-15, 15]^D$	0
F_{11}	Shif. Schaffer	$f_{11}(x) = \sum_{i=1}^{D-1}(x_i^2 + x_{i+1}^2)^{0.25}(\sin^2(50(x_i^2 + x_{i+1}^2)^{0.1}) + 1)$	$[-100, 100]^D$	0

Table 1. Properties of functions F_1–F_{11}

Function	F_{ns}	F'	m_{ns}	Range	$f(x^*)$	Function	F_{ns}	F'	m_{ns}	Range	$f(x^*)$
F_{12}	F_9	F_1	0.25	$[-100, 100]^D$	0	F_{16}	F_9	F_1	0.75	$[-100, 100]^D$	0
F_{13}	F_9	F_3	0.25	$[-100, 100]^D$	0	F_{17}	F_9	F_3	0.75	$[-100, 100]^D$	0
F_{14}	F_9	F_4	0.25	$[-5, 5]^D$	0	F_{18}	F_9	F_4	0.75	$[-5, 5]^D$	0
F_{15}	F_{10}	F_7	0.25	$[-10, 10]^D$	0	F_{19}	F_{10}	F_7	0.75	$[-10, 10]^D$	0

Table 2. Properties of functions F_{12}–F_{19} (functions F_{12}–F_{19} are hybridized by a non-separable function F_{ns} with other function F')

5.2 Parameter Settings

During experimentation, control parameters of the mDE-bcM algorithm are set as follows based on parameter tuning simulation results:

Number of subpopulations is set to 3.

The population size ($NP = 60$) is maintained constant during the evolution process.

DE crossover operator: binomial.

Crossover rate: $CR = 0.9$.

The mDE-bcM and other DE variants were coded in Matlab environment.

The computations were carried out using a PC with Intel(R) Core(TM) i3-2350M @2.3 GHz CPU and 2 GB RAM while running Matlab R2012a on 64-bit Windows operating system.

6 NUMERICAL RESULTS AND DISCUSSIONS

6.1 Simulation Results

The error values ($f(\mathbf{x}) - f(\mathbf{x}^*)$) for all test functions, including the best, median, mean, worst values and standard deviation, are reported in Tables 3 and 4, where $f(\mathbf{x})$ is the global optimum we found, $f(\mathbf{x}^*)$ is the global optimum in Tables 3 and 4, and dimensions $D = 50$, 100, 200, 500 and 1 000. The error values ($f(\mathbf{x}) - f(\mathbf{x}^*)$) was adopted as a performance metric of algorithms. The number of function evaluations (FEs) for each category of dimensions for these problems is set as 30 000.

D	Values	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}
50	Best	4.62E−97	4.04E−44	4.90E+01	0.00E+00	0.00E+00	8.88E−16	1.32E−49	3.94E−81	1.76E−23	1.76E−23
	Median	1.55E−94	5.83E−43	4.90E+01	0.00E+00	0.00E+00	8.88E−16	1.07E−47	2.04E−79	1.24E−22	1.24E−22
	Mean	2.05E−92	8.26E−43	4.90E+01	0.00E+00	0.00E+00	8.88E−16	4.44E−47	8.30E−79	1.87E−22	1.87E−22
	Worst	1.59E−91	3.18E−42	4.90E+01	0.00E+00	0.00E+00	8.88E−16	2.02E−46	6.89E−78	5.58E−22	5.58E−22
	Std	2.71E−93	1.16E−43	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.25E−47	2.92E−79	3.83E−23	3.83E−23
100	Best	7.18E−195	2.13E−86	9.90E+01	0.00E+00	0.00E+00	8.88E−16	2.00E−97	1.06E−159	4.88E−47	0.00E+00
	Median	1.68E−190	1.99E−85	9.90E+01	0.00E+00	0.00E+00	8.88E−16	9.96E−96	1.82E−157	3.34E−46	0.00E+00
	Mean	1.42E−188	3.07E−85	9.90E+01	0.00E+00	0.00E+00	8.88E−16	1.07E−94	6.65E−157	6.52E−46	0.00E+00
	Worst	9.55E−188	1.44E−84	9.90E+01	0.00E+00	0.00E+00	8.88E−16	5.30E−94	8.03E−156	2.07E−45	0.00E+00
	Std	0.00E+00	1.10E−85	0.00E+00	0.00E+00	0.00E+00	0.00E+00	4.04E−95	9.30E−158	1.46E−46	0.00E+00
200	Best	0.00E+00	3.41E−163	1.99E+02	0.00E+00	0.00E+00	8.88E−16	1.52E−161	0.00E+00	0.00E+00	0.00E+00
	Median	0.00E+00	1.05E−162	1.99E+02	0.00E+00	0.00E+00	8.88E−16	4.69E−161	0.00E+00	0.00E+00	0.00E+00
	Mean	0.00E+00	1.03E−162	1.99E+02	0.00E+00	0.00E+00	8.88E−16	4.82E−161	0.00E+00	0.00E+00	0.00E+00
	Worst	0.00E+00	1.55E−162	1.99E+02	0.00E+00	0.00E+00	8.88E−16	9.20E−161	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.64E−162	0.00E+00	0.00E+00	0.00E+00
500	Best	0.00E+00	3.62E−163	4.99E+02	0.00E+00	0.00E+00	8.88E−16	1.51E−161	0.00E+00	0.00E+00	0.00E+00
	Median	0.00E+00	1.07E−162	4.99E+02	0.00E+00	0.00E+00	8.88E−16	5.42E−161	0.00E+00	0.00E+00	0.00E+00
	Mean	0.00E+00	1.05E−162	4.99E+02	0.00E+00	0.00E+00	8.88E−16	5.79E−161	0.00E+00	0.00E+00	0.00E+00
	Worst	0.00E+00	1.55E−162	4.99E+02	0.00E+00	0.00E+00	8.88E−16	1.17E−160	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.68E−162	0.00E+00	0.00E+00	0.00E+00
1 000	Best	0.00E+00	3.87E−163	9.99E+02	0.00E+00	0.00E+00	8.88E−16	1.71E−161	0.00E+00	0.00E+00	0.00E+00
	Median	0.00E+00	1.10E−162	9.99E+02	0.00E+00	0.00E+00	8.88E−16	5.99E−161	0.00E+00	0.00E+00	0.00E+00
	Mean	0.00E+00	1.08E−162	9.99E+02	0.00E+00	0.00E+00	8.88E−16	6.46E−161	0.00E+00	0.00E+00	0.00E+00
	Worst	0.00E+00	1.55E−162	9.99E+02	0.00E+00	0.00E+00	8.88E−16	1.36E−160	0.00E+00	0.00E+00	0.00E+00
	Std	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	9.14E−162	0.00E+00	0.00E+00	0.00E+00

Table 3. Experimental results obtained by mDE-bcM on functions F_1–F_{10}

D	Values	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}	F_{16}	F_{17}	F_{18}	F_{19}
50	Best	7.47E−24	7.80E−23	3.70E+01	3.61E−49	2.16E−23	1.20E+01	3.48E−24	1.17E−45	3.61E−49
	Median	6.55E−23	4.59E−22	3.70E+01	2.66E−47	1.18E−22	1.20E+01	3.19E−23	1.24E−44	2.66E−47
	Mean	1.07E−22	6.56E−22	3.70E+01	1.48E−46	1.69E−22	1.20E+01	4.27E−23	2.48E−44	1.48E−46
	Worst	3.24E−22	1.90E−21	3.70E+01	7.37E−46	4.76E−22	1.20E+01	1.20E−22	1.04E−43	7.37E−46
	Std	2.27E−23	1.19E−22	0.00E+00	5.16E−47	3.23E−23	0.00E+00	8.31E−24	2.36E−45	5.16E−47
100	Best	4.02E−47	7.06E−45	8.70E+01	4.35E−97	8.64E−46	1.20E+01	6.39E−48	3.09E−90	4.35E−97
	Median	4.54E−46	6.65E−44	8.70E+01	2.83E−95	8.34E−45	1.20E+01	7.86E−47	9.52E−89	2.83E−95
	Mean	1.33E−45	1.12E−43	8.70E+01	5.84E−94	1.17E−44	1.20E+01	1.59E−46	3.56E−87	5.84E−94
	Worst	4.74E−45	3.91E−43	8.70E+01	3.19E−93	3.62E−44	1.20E+01	5.11E−46	2.99E−86	3.19E−93
	Std	3.04E−46	2.67E−44	0.00E+00	1.56E−94	2.13E−45	0.00E+00	3.33E−47	3.28E−87	1.56E−94
200	Best	0.00E+00	0.00E+00	1.87E+02	1.28E−161	0.00E+00	1.20E+01	0.00E+00	9.04E−168	1.28E−161
	Median	0.00E+00	0.00E+00	1.87E+02	4.21E−161	0.00E+00	1.20E+01	0.00E+00	1.34E−166	4.21E−161
	Mean	0.00E+00	0.00E+00	1.87E+02	4.39E−161	0.00E+00	1.20E+01	0.00E+00	3.51E−166	4.39E−161
	Worst	0.00E+00	0.00E+00	1.87E+02	8.68E−161	0.00E+00	1.20E+01	0.00E+00	2.05E−165	8.68E−161
	Std	0.00E+00	0.00E+00	0.00E+00	5.56E−162	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.56E−162
500	Best	0.00E+00	0.00E+00	4.87E+02	1.52E−161	0.00E+00	1.20E+01	0.00E+00	1.76E−171	1.52E−161
	Median	0.00E+00	0.00E+00	4.87E+02	5.45E−161	0.00E+00	1.20E+01	0.00E+00	4.05E−170	5.45E−161
	Mean	0.00E+00	0.00E+00	4.87E+02	5.77E−161	0.00E+00	1.20E+01	0.00E+00	7.99E−170	5.77E−161
	Worst	0.00E+00	0.00E+00	4.87E+02	1.17E−160	0.00E+00	1.20E+01	0.00E+00	4.55E−169	1.17E−160
	Std	0.00E+00	0.00E+00	0.00E+00	7.65E−162	0.00E+00	0.00E+00	0.00E+00	0.00E+00	7.65E−162
1 000	Best	0.00E+00	0.00E+00	9.87E+02	1.76E−161	0.00E+00	1.20E+01	0.00E+00	4.66E−173	1.76E−161
	Median	0.00E+00	0.00E+00	9.87E+02	5.85E−161	0.00E+00	1.20E+01	0.00E+00	9.61E−172	5.85E−161
	Mean	0.00E+00	0.00E+00	9.87E+02	6.35E−161	0.00E+00	1.20E+01	0.00E+00	6.07E−171	6.35E−161
	Worst	0.00E+00	0.00E+00	9.87E+02	1.36E−160	0.00E+00	1.20E+01	0.00E+00	4.19E−170	1.36E−160
	Std	0.00E+00	0.00E+00	0.00E+00	9.33E−162	0.00E+00	0.00E+00	0.00E+00	0.00E+00	9.33E−162

Table 4. Experimental results obtained by mDE-bcM on functions F_{11}–F_{19}

Tables 3 and 4 show that there are 11 functions where median values are equal to 0, then 4 functions whose median values are less than 10E−160, and 4 functions where median values are worse than 10E−160 in 19 functions, when $D = 1\,000$. But when $D = 50$, there are 2 functions whose median values are equal to 0, and 17 functions whose median values are worse than 10E−160, what shows that the algorithm mDE-bcM performs well in solving high-dimensional problems.

Tables 3 and 4 also show that there are 10 functions whose median values are equal to 0, 5 functions whose median values are less than 10E−160, and 4 functions whose median values are worse than 10E−160 in 19 functions, when $D > 200$, which shows that the algorithm mDE-bcM performs well when the dimension D exceeds 200.

But Tables 3 and 4 also reveal that the mDE-bcM appears to have difficulties on function F_3 (Shifted Rosenbrock's function), F_5 (Shifted Griewank's function), F_6 (Shifted Ackley's function), F_{13} (hybrid composition function) and F_{16} (hybrid composition function), regardless of the dimension. Functions F_3 and F_6 are multimodal functions, functions F_{13} and F_{16} are hybrid composition functions and hybridized by a non-separable function F_9 with other function F_3 or F_1. This shows that the algorithm mDE-bcM performs poorly on some multimodal functions, although it performs well on multimodal function F_4. If it is used in multimodal function optimization, premature convergence must be avoided.

In Table 3 and 4, the mean error is better than its corresponding median, which means that individuals are evenly distributed in the population, without much worse values.

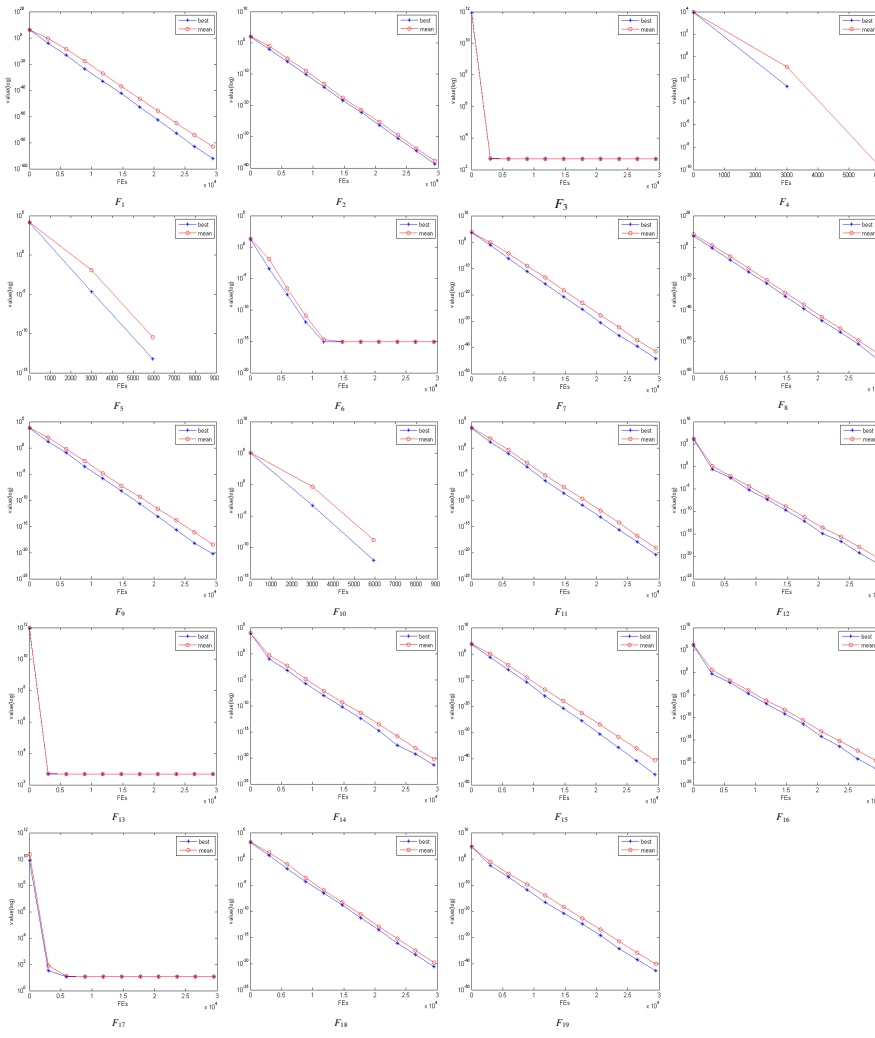


Figure 1. The best value and the mean value, where dimension $D = 500$, the test functions are from F_1 to F_{19}

The best values and the mean values of 19 test functions are shown in Figure 1, where dimension $D = 500$, number of iteration is 30 000. The horizontal axis is the number of iterations (FEs), and the vertical axis is the error values of fitness (log).

From Figure 1, the continuous optimization process of the mDE-bcM on functions F_1–F_{19} can be observed, showing that the mDE-bcM has excellent performance on most functions. But the best error and mean error were rapidly reduced in the

early stages of iteration and slowly changed in the later stages of iteration on F_3, F_6, F_{13} and F_{17}, uniformly reduced on other functions. By analyzing individuals in population, we observed that rapid convergence occurred on F_3, F_6, F_{13} and F_{17}, approximate optimal solutions are found after about 300 generations, and after that, improvements are very slow, and more generations only consume the processing time, what shows that the mDE-bcM has general performance on these functions. In fact, a single mutation strategy always performs poorly.

In addition, the curve is interrupted on F_4, F_5 and F_{10}, because the error values $(f(\mathbf{x}) - f(\mathbf{x}^*))$ are less than 0 and the vertical axis is logarithmic.

Overall, the mDE-bcM was able to optimize on 2 functions at $D = 50$, 3 functions at $D = 100$, 11 functions at $D = 200$, 11 functions at $D = 500$, and 11 functions at $D = 1\,000$. It was also able to obtain high quality results for the rest of the functions at different dimensions with small errors.

In summary, the mDE-bcM has excellent performance on most functions, especially in high dimension, although it encounters difficulties in a few functions.

6.2 Parameter Tuning

In this subsection, we will evaluate the sensitivity of the proposed algorithm to some parameters. The decision variables to be adjusted in mDE-bcM include the population size (NP) and mutation strategies.

In these tests, we varied NP at a time while keeping the other parameters fixed, or used different strategies in the evolutionary process. We performed 25 independent runs for every set of parameters. In order to test the effect of the population size (NP) and mutation strategies more clearly, the population was not grouped, only one population was used.

Table 5 shows results of parameter tuning while population size NP changes from 30 to 300, where $D = 500$. Figure 2 shows comparison between our best-and-current mutation strategy and 7 classic mutation strategies, where $NP = 60$, $D = 500$.

NP	30	60	90	180	300
F_1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.28E−189
F_3	4.99E+02	4.99E+02	4.99E+02	4.99E+02	4.99E+02
F_4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_7	1.41E−161	1.51E−161	1.73E−161	3.95E−158	4.87E−95
F_8	0.00E+00	0.00E+00	0.00E+00	1.27E−258	3.34E−155
F_{15}	1.08E−161	1.52E−161	1.60E−161	4.27E−159	1.01E−94
F_{19}	2.45E−169	1.76E−171	6.06E−172	5.02E−154	7.32E−94

Table 5. Population size (NP) tuning. The data in Table 5 are the average of 25 calculations, where $D = 500$.

Table 5 exhibits different errors during the fitness evaluations while the mDE-bcM selects different population size NP, it is obvious that better results can be

obtained where $NP < 90$, and optimal results can be obtained on most functions while $NP = 60$. Bigger population size makes it easier for individuals in the population to carry more abundant genes, maintain the diversity of the population and quickly obtain high-quality solutions, but it also means that more extra fitness evaluations are done, what consumes more computing time. A smaller population size means shorter computing time, but it leads to poor population diversity and calculation success rate. Although the mDE-bcM can also obtain better results on some functions while $NP = 30$, as a compromise, we select $NP = 60$ in this paper, and sub-population size is 20 after grouping.

The mean values of our best-and-current mutation strategy and other 7 mutation strategies on functions F_1, F_2, F_6, F_7, F_8 and F_9 are shown in Figure 2.

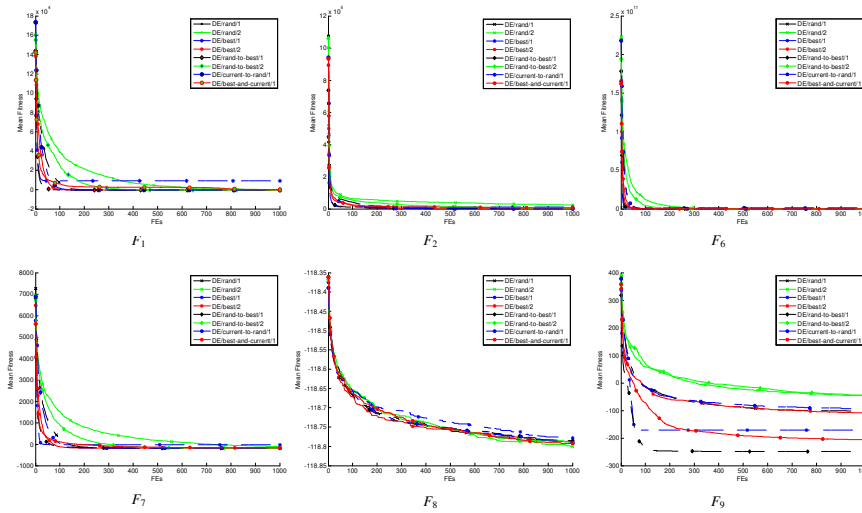


Figure 2. Comparisons between our best-and-current mutation strategy and 7 mutation strategies on functions F_1, F_2, F_6, F_7, F_8 and F_9

Figure 2 illustrates that our best-and-current mutation strategy performs better than most traditional mutation strategies, because its base vector is linear combination of the best vector $\mathbf{x}_{\text{best},g}$ and current vector $\mathbf{x}_{i,g}$, instead of the best vector $\mathbf{x}_{\text{best},g}$ or the random/current vector $\mathbf{x}_{i,g}$, so it inherits two excellent genes of the parent as base vectors at the same time. Like other methods, best-and-current mutation strategy also converges rapidly on functions F_1, F_2, F_6 and F_7, that means the mDE-bcM and other mutation strategies fall into prematurity and individuals become the same, test results on 19 functions also show that a single mutation strategy will always perform poorly on some functions, so it is necessary to adopt multiple mutation strategies or combinations of them.

6.3 Computational Complexity Analysis

The mDE-bcM divides initial population X_0 into three subpopulations S1, S2 and S3, subpopulation size is reduced to $NP/3$, then mDE-bcM carries out the evolutionary operations such as crossover, mutation and selection in parallel, but it is still a serial algorithm structure between two runs, so computational complexity is determined by the number of calls to genetic operators, as shown in the following [27, 28]:

$$O(D \cdot NP/3 \cdot G_m). \tag{25}$$

In mDE-bcM, the outer loop controls the number of iterations and its maximum value is G_m, and the inner loop controls the number of individuals involved in evolution and its maximum value is $NP/3$, each individual contains D components. In formula (25), D and G_m are necessary for high dimension and high precision, so reducing the number of individuals from NP to $NP/3$ can effectively reduce the computational complexity, which is the value of multiple populations with parallel computing.

6.4 Comparison with Other State-of-the-Art Optimization Techniques

In this section, we compared mDE-bcM with two groups of state-of-the-art optimization algorithms, which is because these two groups of algorithms were tested on SOCO 2011 [25, 26]. Like the analysis in literature [29], different algorithms have different performance in different test function suites, no algorithm is always excellent in all test function suites. For example, MOS-CEC2013 [30] performs best on the CEC 2010 benchmark suite [31] and CEC 2013 benchmark suite [32], but ranks 6$^{\text{th}}$ on the SOCO 2011 benchmark suite, MOS-SOCO2011 [33] performs best on the CEC 2010 benchmark suite, but ranks 4$^{\text{th}}$ on the SOCO 2011 benchmark suite and 8$^{\text{th}}$ CEC 2013 benchmark suite. On the other hand, the algorithms from the first group are all tested in different dimensions, which can reflect the ability of the algorithm to deal with large-scale optimization problems in different dimensions, whereas the algorithms from the second group are all tested in the dimension of 1 000, which only reflects the high-dimensional processing ability of the algorithm.

For this reason, both groups of algorithms were tested by researchers on 19 benchmark large-scale global continuous optimization functions, these functions were taken from the special issue of Soft Computing on scalability of evolutionary algorithms [25, 26], the algorithms from the first group were compared in [11], whereas the algorithms from the second group were compared in [34].

We have also used the Friedman test for multiple comparisons to check differences among the considered algorithms, the statistical methods include:

1. Overall ranking according to the Friedman test: we firstly rank each algorithm according to its mean value for each function, then compute its average ranking on all the functions.

2. # Best: This is the number of functions for which each algorithm obtains the best results compared to all the other algorithms.
3. nWins: This is the number of our mDE-bcM is better than, similar to and worse than that of the corresponding algorithm according to the Wilcoxon Signed Rank Test in a pair-wise comparison [35].
4. The Friedman test: The Friedman test can detect whether there are significant differences between the behavior of multiple algorithms.

The comparison algorithms in the first group are as follows:

1. The classic DE algorithm [1]. The strategy is "DE/rand/1/exp", $CR = 0.9$, $F = 0.5$.
2. Real-coded Genetic Algorithm (CHC) [36]. It is a real-coded genetic algorithm using interval-schemata as an analysis tool and was tested by 13 test functions.
3. CMA-ES [37]. It introduced a restart-CMA evolution strategy and was evaluated on 25 test functions of the CEC 2005 whose dimension $D = 10$, 30 and 50. In CMA-ES, the default population size prescribed for the (μ_W, λ)-CMA-ES grew with $logD$ and equaled to $\lambda = 10$, 14, 15 for $D = 10$, 30, 50, respectively. On multi-modal functions, the optimal population size λ could be considerably greater than the default population size.
4. MA-SSW [38]. Molina et al. proposed a memetic algorithm based on local search chains for high dimensionality, MA-SSW-Chains used the Subgrouping Solis Wets' algorithm as its local search method, in which only a random subset of the variables was explored and this subset would change after a certain number of evaluations. MA-SSW-Chains is considered to be one of the best algorithms in CEC 2010 benchmark suite.
5. MTS-LS1 [39]. It introduced multiple agents to search the solution space and was evaluated on 7 test functions of the CEC 2008 special session and competition on large scale global optimization. Each agent in MTS-LS1 did an iterated local search using one of three candidate local search methods and might find its way to a local optimum or the global optimum. MOS-SOCO2011 based on MTS-LS1 is considered to be one of the best algorithms in SOCO 2011 benchmark suite.
6. SaDE [7]. In SaDE, the trial vector generation strategies with their associated parameters values were gradually self-adapted by learning from previous successful experiences. SaDE was evaluated on 26 test functions, two of which were chosen from [40].
7. EvoPR [41]. Rahnamayan et al. proposed the evolutionary path relinking (EvoPR) to finding a global optimum of multi-modal functions and unconstrained large-scale problems, EvoPR was tested on 19 test functions for the special issue of Soft Computing on scalability of evolutionary algorithms and

other metaheuristics for large scale continuous optimization problems [25] with dimensions ranging from 50 to 1 000.

8. mDE-bES [11]. The mDE-bES utilized exponential crossover, and divided the population into four subgroups, each of which employed a certain modified mutation strategy, $CR \in [0.2, 0.9]$ and $F \in [0.5, 0.9]$.

The algorithms from the first group were initially tested on different functions, but were finally tested on the same initial population, same benchmark suite $F1$–$F19$ and same stopping rule in [11], and dimension changed from 50 to 1 000, but G-CMA-ES is not evaluated at dimension 1 000.

The comparison algorithms in the second group are from literature [29, 34]. [29] provided a comprehensive comparison of the performance of several algorithms evaluated on the CEC 2010, CEC 2013 and SOCO 2011 benchmark suites, [34] compared its algorithm CCJADE with six state-of-the-art algorithms, according to their analysis, the algorithms with excellent performance in recent years are as follows:

1. MOS-CEC2013 [30]. MOS-CEC2013 is a hybrid algorithm that combines a population-based search algorithm, a Genetic Algorithm (GA), with two powerful local searches (the Solis Wets' algorithm and a variation of the MTS-LS1 local search). It was the champion of the competition on LSGO used the CEC 2013 LSGO benchmark suite in 2013 which defined 15 test functions with dimension 1 000 or 905 in [32].
2. MOS-SOCO2011 [29, 33]. MOS-SOCO2011 combined a Differential Evolution (DE) algorithm and the first of the local searches of the MTS algorithm (MTS-LS1), and used the Multiple Offspring (MOS) framework which made the seamless combination of multiple search algorithm in a dynamic way. It obtained the best overall results among all the algorithms included in the 2011 special issue of the Soft Computing journal.
3. jDElscop [42]. The jDElscop was a self-adaptive DE algorithm that used three different DE strategies (DE/rand/1/bin, DE/rand/1/exp and DE/best/1/bin), a sign-change control mechanism for the F parameter and a new population size reduction mechanism. It was the runner-up in the 2011 special issue of the Soft Computing journal.
4. GaDE [43]. GaDE was a generalization of the adaptive DE algorithm, which used a probability distribution to adapt the value of each of the parameters of the algorithm for each of the individuals of the population. It obtained the third place among all the algorithms included in 2011 special issue of the Soft Computing journal.
5. DECC-G [44]. The DECC-G was a cooperative coevolution algorithm, which divided large problems into small components that were optimized independently by certain EAs. It won the second place in the competition on LSGO in 2013 and the fourth place in 2015.

6. CCJADE [34]. The CCJADE used a surrogate-assisted CC (SACC) optimizer, in which fitness surrogates are exploited within the low-dimensional subcomponents resulting from the problem decomposition, and it was tested on the SOCO 2011 benchmark suite where $D = 1\,000$.

7. L-SHADE [45]. The L-SHADE further extended SHADE with Linear Population Size Reduction (LPSR), which continually decreased the population size according to a linear function, and L-SHADE was evaluated on CEC2014 benchmarks.

8. LM-CMA-ES [46]. The LM-CMA-ES was a computationally efficient limited memory Covariance Matrix Adaptation Evolution Strategy for large scale optimization, which sampled candidate solutions according to a covariance matrix reproduced from m direction vectors selected during the optimization process, the decomposition of the covariance matrix into Cholesky factors could reduce the time and memory complexity of the sampling. LM-CMA-ES could efficiently solve fully non-separable problems and reduce the overall run-time.

The algorithms from the second group were initially tested on different functions, but were finally tested on the same functions at dimension 1 000 in [34].

Because of the difference of the running environment, such as hardware platform, programming language and program efficiency, etc., the performance of different algorithms is very different, and the test results in different environments are actually not comparable. Therefore, the test results of the algorithms in the first group are coming from [11], the test results of the algorithms in the second group are coming from [34], because they give the best results that these algorithms can get.

The comparison results of the mDE-bcM and the first group of algorithms is shown in Tables 6, 7, 8, 9 and 10, we run for 25 times on every function $F_1 - F_{19}$, where D changes from 50 to 1 000. Results highlighted in boldface show the best mean values for each function. As suggested in the special issue [25], all values below 1.00E−14 are approximated to 0.00E+00.

Tables 6, 7, 8, 9 and 10 show that the proposed mDE-bcM has superior performance compared with the first group of algorithms on functions $F_1 - F_{19}$, while D changes from 50 to 1 000. As indicated in Tables 6, 7 and 8, the mDE-bcM can obtain optima except functions F_3, F_{13} and F_{17}, where D changes from 50 to 200. In Tables 9 and 10, the mDE-bcM can obtain optima except functions F_3 and F_{13}, where D changes from 500 to 1 000. The number of times the mDE-bcM can obtain the optimal solution is higher than that of other algorithms.

	DE	CHC	CMA-ES	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
F_1	**0.00E+00**	1.67E−11	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.68E+01	1.22E−02	**0.00E+00**	**0.00E+00**
F_2	8.84E−11	6.19E+01	2.75E−11	7.61E−02	8.84E−14	1.21E+02	3.71E−01	1.52E+01	**0.00E+00**
F_3	1.63E+02	1.25E+06	7.97E−01	4.79E+01	1.63E+02	7.46E+04	1.12E+02	**4.76E−05**	4.90E+01
F_4	**0.00E+00**	7.43E+01	1.05E+02	r1.19E−01	**0.00E+00**	1.07E+01	4.96E−02	1.77E+01	**0.00E+00**
F_5	7.68E−03	1.67E−03	2.96E−04	**0.00E+00**	7.68E−03	1.87E−01	5.13E−02	**0.00E+00**	**0.00E+00**
F_6	**0.00E+00**	6.15E−07	2.09E+01	4.89E−14	**0.00E+00**	4.63E−02	6.85E−03	3.97E−14	**0.00E+00**
F_7	**0.00E+00**	2.66E−09	1.01E−10	**0.00E+00**	**0.00E+00**	**0.00E+00**	2.63E−02	**0.00E+00**	**0.00E+00**
F_8	9.56E−12	2.24E+02	**0.00E+00**	3.06E−01	9.65E−12	6.92E+05	2.08E+02	1.64E−09	**0.00E+00**
F_9	1.03E+02	3.10E+02	1.66E+01	2.94E+02	1.03E+02	3.00E−02	8.02E+00	**0.00E+00**	**0.00E+00**
F_{10}	**0.00E+00**	7.30E+00	6.81E+00	**0.00E+00**	**0.00E+00**	2.94E−02	4.80E−02	**0.00E+00**	**0.00E+00**
F_{11}	1.04E+02	2.16E+00	3.01E+01	r4.49E−03	1.04E+02	8.35E−02	9.68E+00	1.15E−08	**0.00E+00**
F_{12}	1.34E+01	9.57E−01	1.88E+02	**0.00E+00**	1.34E+01	4.80E+01	r2.27E+00	**0.00E+00**	**0.00E+00**
F_{13}	2.94E+01	2.08E+06	1.97E+02	3.02E+01	2.94E+01	3.42E+09	4.22E+01	**2.50E−01**	3.70E+01
F_{14}	5.52E+01	6.17E+01	1.09E+02	**0.00E+00**	5.52E+01	4.22E+03	9.97E−01	9.60E+00	**0.00E+00**
F_{15}	**0.00E+00**	3.98E−01	9.79E−04	**0.00E+00**	**0.00E+00**	8.50E−03	6.38E−02	**0.00E+00**	**0.00E+00**
F_{16}	4.06E+01	2.95E−09	4.27E+02	4.06E−03	4.06E+01	1.36E+01	5.63E+00	**0.00E+00**	**0.00E+00**
F_{17}	2.17E+02	2.26E+04	6.89E+02	2.60E+01	2.17E+02	2.36E+05	6.77E+01	2.42E−01	1.20E+01
F_{18}	5.65E+01	1.58E+01	1.31E+02	**0.00E+00**	5.65E+01	2.72E+01	1.62E+00	5.65E−05	**0.00E+00**
F_{19}	**0.00E+00**	3.59E+02	4.76E+00	**0.00E+00**	**0.00E+00**	1.15E−01	5.03E−02	**0.00E+00**	**0.00E+00**

Table 6. Comparison of mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 50$

	DE	CHC	CMA-ES	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
F_1	3.79E+00	3.56E−11	**0.00E+00**	**0.00E+00**	1.09E−12	3.13E+01	4.34E−02	**0.00E+00**	**0.00E+00**
F_2	7.58E+01	8.58E+01	1.51E−10	7.01E+00	4.66E−10	1.26E+02	3.30E+00	4.00E+01	**0.00E+00**
F_3	1.27E+02	4.19E+06	3.88E+00	1.38E+02	2.32E+02	1.11E+05	3.98E+02	**4.90E−01**	9.90E+01
F_4	2.85E+00	2.19E+02	2.50E+02	1.19E−01	1.05E−12	1.58E+01	1.07E−01	1.87E+01	**0.00E+00**
F_5	3.05E−01	3.83E−03	1.58E−03	**0.00E+00**	6.70E−03	3.53E−01	3.92E−02	**0.00E+00**	**0.00E+00**
F_6	4.34E−01	4.10E−07	2.12E+01	6.03E−14	1.20E−12	8.32E−02	2.50E−04	1.44E−13	**0.00E+00**
F_7	**0.00E+00**	1.40E−02	4.22E−04	**0.00E+00**	**0.00E+00**	**0.00E+00**	9.17E−02	**0.00E+00**	**0.00E+00**
F_8	4.74E+02	1.69E+03	**0.00E+00**	3.48E+01	1.43E−03	2.83E+05	2.27E+03	2.32E−03	**0.00E+00**
F_9	3.71E−03	5.86E+02	1.02E+02	5.63E+02	2.20E+02	3.00E−02	2.91E+01	**0.00E+00**	**0.00E+00**
F_{10}	**0.00E+00**	3.30E+01	1.66E+01	**0.00E+00**	**0.00E+00**	4.73E−02	2.05E−01	**0.00E+00**	**0.00E+00**
F_{11}	8.58E−04	7.32E+01	1.64E+02	1.09E−01	2.10E+02	3.05E−01	2.60E+01	**0.00E+00**	**0.00E+00**
F_{12}	2.71E+00	1.03E+01	4.17E+02	3.28E−03	3.91E+01	3.79E+01	5.01E+00	5.36E−04	**0.00E+00**
F_{13}	5.87E+01	2.70E+06	4.21E+02	8.35E+01	1.75E+02	3.42E+09	1.40E+02	**8.50E+00**	8.70E+01
F_{14}	2.21E+00	1.66E+02	2.55E+02	**0.00E+00**	2.04E+02	3.92E+03	1.24E+00	1.16E+01	**0.00E+00**
F_{15}	**0.00E+00**	8.13E+00	6.30E−01	**0.00E+00**	**0.00E+00**	3.99E−02	6.56E−02	**0.00E+00**	**0.00E+00**
F_{16}	3.52E+00	2.23E+01	8.59E+02	1.61E−02	1.04E+02	1.96E+01	8.29E+00	**0.00E+00**	**0.00E+00**
F_{17}	1.58E+01	1.47E+05	1.51E+03	9.92E+01	4.17E+02	2.34E+05	1.97E+02	**6.65E−03**	1.20E+01
F_{18}	8.76E−01	7.00E+01	3.07E+02	**0.00E+00**	1.22E+02	3.05E+01	3.34E+00	4.46E−01	**0.00E+00**
F_{19}	**0.00E+00**	5.45E+02	2.02E+01	**0.00E+00**	**0.00E+00**	2.71E−01	1.43E−01	**0.00E+00**	**0.00E+00**

Table 7. Comparison of mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 100$

With the increase of dimension, the mDE-bcM has more obvious advantages, the number of functions on which it can obtain the optimal value is increasing, while the ability of other algorithms to obtain the optimal value is decreasing. Therefore, we can conclude that mDE-bcM is more suitable for dealing with large-scale high-dimension optimization problems.

The average ranking, the number of functions with best results and the nWins value for mDE-bcM and the first group of algorithms are reported in Table 11. Limited to space, we only give the statistical results at dimension 1 000.

	DE	CHC	CMA-ES	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
F_1	8.55E+00	8.34E−01	**0.00E+00**	**0.00E+00**	2.29E+00	2.03E+01	8.03E−02	**0.00E+00**	**0.00E+00**
F_2	1.05E+02	1.03E+02	1.16E−09	3.36E+01	4.54E−09	1.03E+02	8.03E+00	4.15E+01	**0.00E+00**
F_3	3.32E+05	2.01E+07	**8.91E+01**	2.50E+02	1.69E+02	4.82E+04	2.91E+02	1.35E+02	1.99E+02
F_4	6.98E+00	5.40E+02	6.48E+02	4.43E+00	2.34E−12	6.25E+00	3.52E−01	9.27E−13	**0.00E+00**
F_5	4.05E−01	8.76E−03	**0.00E+00**	**0.00E+00**	5.42E−03	6.43E−02	2.68E−02	**0.00E+00**	**0.00E+00**
F_6	7.14E−01	1.23E+00	2.14E+01	1.19E−13	2.38E−12	2.73E−02	6.22E−01	**0.00E+00**	**0.00E+00**
F_7	**0.00E+00**	2.59E−01	1.17E−01	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.82E−02	**0.00E+00**	**0.00E+00**
F_8	5.76E+03	9.38E+03	**0.00E+00**	7.23E+02	1.42E+01	4.47E+05	1.34E+04	8.71E−01	**0.00E+00**
F_9	8.79E−03	1.19E+03	3.75E+02	1.17E+03	4.27E+02	3.00E−02	6.22E+01	**0.00E+00**	**0.00E+00**
F_{10}	4.19E−02	7.13E+01	4.43E+01	**0.00E+00**	**0.00E+00**	1.59E−02	1.04E+00	**0.00E+00**	**0.00E+00**
F_{11}	5.07E−03	3.85E+02	8.03E+02	3.50E−01	4.28E+02	4.89E−03	5.93E+01	**0.00E+00**	**0.00E+00**
F_{12}	3.61E+00	7.44E+01	9.06E+02	1.75E−02	8.42E+01	4.63E+01	1.00E+01	**0.00E+00**	**0.00E+00**
F_{13}	1.49E+02	5.75E+06	9.43E+02	1.68E+02	2.53E+02	3.16E+09	1.71E+02	**9.45E+01**	1.87E+02
F_{14}	4.75E+00	4.29E+02	6.09E+02	9.76E−01	3.98E+02	4.09E+03	3.75E+00	1.20E+01	**0.00E+00**
F_{15}	**0.00E+00**	2.14E+01	1.75E+00	**0.00E+00**	**0.00E+00**	5.38E−03	3.80E−01	**0.00E+00**	**0.00E+00**
F_{16}	3.70E+00	1.60E+02	1.92E+03	6.02E−02	1.97E+02	9.49E+00	1.74E+01	**0.00E+00**	**0.00E+00**
F_{17}	2.23E+01	1.75E+05	3.36E+03	7.55E+01	6.07E+02	2.36E+05	1.56E+02	8.39E−02	1.20E+01
F_{18}	2.37E+00	2.12E+02	6.89E+02	4.29E−04	2.34E+02	1.69E+01	8.85E+00	8.93E−11	**0.00E+00**
F_{19}	4.19E−02	2.06E+03	7.52E+02	**0.00E+00**	**0.00E+00**	1.00E−01	2.15E+00	**0.00E+00**	**0.00E+00**

Table 8. Comparison of mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 200$

	DE	CHC	CMA-ES	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
F_1	2.46E+01	2.84E−12	**0.00E+00**	**0.00E+00**	5.77E−12	1.34E+01	**0.00E+00**	3.92E−13	**0.00E+00**
F_2	1.44E+02	1.29E+02	3.48E−04	7.86E+01	5.34E−06	9.23E+01	2.04E+01	4.56E+01	**0.00E+00**
F_3	1.12E+05	1.14E+06	3.58E+02	6.07E+02	2.20E+02	2.62E+04	5.97E+02	**4.16E+02**	4.99E+02
F_4	1.63E+01	1.91E+03	2.10E+03	1.78E+02	5.62E−12	1.31E+00	1.45E+00	1.91E−11	**0.00E+00**
F_5	4.73E−01	6.98E−03	2.96E−04	**0.00E+00**	4.24E−03	7.48E−03	3.03E−02	1.83E−13	**0.00E+00**
F_6	1.06E+00	5.16E+00	2.15E+01	2.63E−13	6.18E−12	4.63E−01	1.21E+00	3.56E−14	**0.00E+00**
F_7	**0.00E+00**	1.27E−01	7.21E+153	4.69E−14	1.46E−12	**0.00E+00**	8.06E−03	**0.00E+00**	**0.00E+00**
F_8	6.70E+04	7.22E+04	2.36E−06	1.32E+04	6.16E+03	3.21E+05	7.05E+04	5.48E+02	**0.00E+00**
F_9	1.12E−02	3.00E+03	1.74E+03	2.53E+03	1.00E+03	3.00E−02	1.75E+02	**0.00E+00**	**0.00E+00**
F_{10}	2.93E−01	1.86E+02	1.27E+02	2.80E−01	**0.00E+00**	8.41E−03	3.29E+01	**0.00E+00**	**0.00E+00**
F_{11}	2.43E−01	1.81E+03	4.16E+03	4.21E+01	1.00E+03	2.22E−03	1.77E+02	**0.00E+00**	**0.00E+00**
F_{12}	1.16E+01	4.48E+02	2.58E+03	2.55E+01	2.47E+02	4.61E+01	1.73E+01	**0.00E+00**	**0.00E+00**
F_{13}	4.02E+02	3.22E+07	2.87E+03	4.00E+02	5.05E+02	2.97E+09	5.75E+02	**3.23E+02**	4.87E+02
F_{14}	1.16E+01	1.46E+03	1.95E+03	5.65E+01	1.10E+03	3.91E+03	9.00E+00	1.68E+01	**0.00E+00**
F_{15}	4.19E−02	6.01E+01	2.82E+262	5.53E+00	1.08E−12	2.84E−03	2.25E+00	**0.00E+00**	**0.00E+00**
F_{16}	1.32E+01	9.55E+02	5.45E+03	1.08E−01	4.99E+02	5.82E+00	4.87E+01	**0.00E+00**	**0.00E+00**
F_{17}	6.94E+01	8.40E+05	9.59E+03	1.38E+02	7.98E+02	2.38E+05	3.94E+02	6.65E+01	**1.20E+01**
F_{18}	3.87E+00	7.32E+02	2.05E+03	2.41E−03	5.95E+02	9.43E+00	3.28E+01	**0.00E+00**	**0.00E+00**
F_{19}	8.39E−02	1.76E+03	2.44E+06	**0.00E+00**	**0.00E+00**	1.00E−01	5.00E+01	**0.00E+00**	**0.00E+00**

Table 9. Comparison of mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 500$

In Table 11, the algorithm that obtains the best ranking, number of functions with the best results and the nWins value is mDE-bcM, followed by mDE-bES. The average ranking of mDE-bcM is 1.21, the numbers of functions that mDE-bcM is better than that of DE, CHC, MA-SSW, MTS-LS1, SaDE, EvoPR, mDE-bES are 18, 19, 18, 13, 18, 19, 8, respectively. This shows that mDE-bcM has obviously good optimization performance.

The Friedman test reported a p-value = 9.42E−14 for Table 10, which is below the significance level $\alpha = 0.05$, and it means that there are statistical differences

	DE	CHC	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
F_1	3.71E+01	1.36E−11	**0.00E+00**	1.15E−11	3.49E+01	4.00E−05	8.24E−13	**0.00E+00**
F_2	1.63E+02	1.44E+02	1.39E+02	2.25E−02	1.43E+02	3.21E+01	5.97E+01	**0.00E+00**
F_3	1.59E+05	8.75E+03	1.22E+03	**2.10E+02**	1.62E+05	1.12E+03	9.00E+02	9.99E+02
F_4	3.47E+01	4.76E+03	1.58E+03	1.15E−11	3.21E+01	4.08E+02	4.03E+01	**0.00E+00**
F_5	7.36E−01	7.02E−03	5.92E−04	3.55E−03	6.33E−01	3.72E−02	**0.00E+00**	**0.00E+00**
F_6	8.70E−01	1.38E+01	1.46E−09	1.24E−11	4.28E−01	1.97E+00	1.28E−12	**8.88E−16**
F_7	**0.00E+00**	3.52E−01	6.23E−13	**0.00E+00**	**0.00E+00**	1.50E−04	**0.00E+00**	**0.00E+00**
F_8	3.15E+05	3.11E+05	7.49E+04	1.23E+05	3.08E+05	2.15E+05	7.98E+03	**0.00E+00**
F_9	6.26E−02	6.11E+03	5.99E+03	1.99E+03	3.00E−02	4.07E+02	**0.00E+00**	**0.00E+00**
F_{10}	1.67E−01	3.83E+02	2.09E−05	**0.00E+00**	1.47E−01	3.86E+02	**0.00E+00**	**0.00E+00**
F_{11}	4.42E−02	4.82E+03	5.27E+01	1.99E+03	4.56E−01	3.96E+02	**0.00E+00**	**0.00E+00**
F_{12}	2.58E+01	1.05E+03	9.48E−02	5.02E+02	3.43E+01	3.23E+01	**0.00E+00**	**0.00E+00**
F_{13}	8.24E+04	6.66E+07	1.02E+03	8.87E+02	3.27E+09	1.13E+03	**6.34E+02**	9.87E+02
F_{14}	2.39E+01	3.62E+03	7.33E+02	2.23E+03	3.71E+03	4.31E+02	2.45E+01	**0.00E+00**
F_{15}	2.11E−01	8.37E+01	1.16E−13	**0.00E+00**	1.11E−01	1.26E+02	**0.00E+00**	**0.00E+00**
F_{16}	1.83E+01	2.32E+03	2.19E+00	1.00E+03	2.37E+01	8.44E+01	**0.00E+00**	**0.00E+00**
F_{17}	1.76E+05	2.04E+07	3.26E+02	1.56E+03	1.62E+05	6.75E+02	1.88E+02	**1.20E+01**
F_{18}	7.55E+00	1.72E+03	2.58E+01	1.21E+03	3.54E+01	1.95E+02	2.49E−01	**0.00E+00**
F_{19}	2.51E−01	4.20E+03	1.56E−12	**0.00E+00**	9.32E−01	2.03E+02	**0.00E+00**	**0.00E+00**

Table 10. Comparison of mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 1\,000$

	DE	CHC	MA-SSW	MTS-LS1	SaDE	EvoPR	mDE-bES	mDE-bcM
Ranking	4.58	6.37	3.42	3.42	4.79	4.79	1.79	1.21
# Best	1	0	1	5	1	0	10	17
nWins-better	18	19	18	13	18	19	8	–
nWins-similar	1	0	1	4	1	0	9	–
nWins-worse	0	0	0	2	0	0	2	–
p-value	2.21E−05	1.31E−05	2.21E−05	4.50E−03	2.21E−05	1.31E−05	1.14E−02	–

Table 11. Average ranking, number of functions with the best results, nWins value and the Friedman test for mDE-bcM and the first group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 1\,000$

as the negation of the null hypothesis. Used mDE-bcM as the control algorithm, further the Friedman tests show that mDE-bcM have been significant differences with other 7 algorithms.

The comparison results of mDE-bcM and the second group of algorithms are in Table 12, we run for 25 times on every function $F_1 - F_{19}$, where $D = 1\,000$, except for our mDE-bcM, FEs is 5×10^6.

The average ranking, the number of functions with best results and the nWins value for mDE-bcM and the second group of algorithms are reported in Table 13.

In Table 13, the algorithms that obtain the best ranking are mDE-bcM and MOS-SOCO2011, number of functions with the best results is mDE-bcM, followed by MOS-SOCO2011, and the numbers of functions that mDE-bcM is better than that of MOS-CEC2013, MOS-SOCO2011, jDElscop, GaDE, DECC-G, CCJADE, L-SHADE and LM-CMA-ES are 14, 3, 9, 15, 15, 12, 18 and 18, respectively. This shows that mDE-bcM and MOS-SOCO2011 have the same performance, but they have significant advantages over other algorithms.

	MOS-CEC2013	MOS-SOCO2011	jDElscop	GaDE	DECC-G	CCJADE	L-SHADE	LM-CMA-ES	mDE-bcM
F_1	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	3.26E−06	1.80E−15	7.82E−04	2.16E−13	**0.00E+00**
F_2	1.10E+02	5.88E−01	2.46E+01	5.46E+01	1.31E+03	1.38E+02	4.55E+01	1.49E+02	**0.00E+00**
F_3	7.39E+00	7.09E+01	8.51E+02	9.47E+02	**1.09E+00**	3.74E+02	1.64E+03	6.04E+02	9.99E+02
F_4	**0.00E+00**	**0.00E+00**	2.39E−01	3.79E−02	2.16E+11	8.62E−01	1.71E+03	1.62E+04	**0.00E+00**
F_5	**0.00E+00**	**0.00E+00**	0.00E+00	5.91E−04	8.30E+06	4.85E−04	6.07E−03	1.13E−13	**0.00E+00**
F_6	**0.00E+00**	**0.00E+00**	1.16E−12	2.55E−14	9.63E−01	3.28E−13	1.92E+00	1.99E+01	8.88E−16
F_7	2.56E−12	**0.00E+00**	**0.00E+00**	**0.00E+00**	Inf	**0.00E+00**	1.34E+00	4.22E+569	**0.00E+00**
F_8	5.98E+03	1.66E+05	3.17E+04	1.55E+04	1.11E+05	1.92E+06	5.76E+04	1.66E−06	**0.00E+00**
F_9	2.51E+03	0.00E+00	9.21E−08	4.29E−04	1.78E+01	9.47E−01	1.84E+03	9.17E+03	**0.00E+00**
F_{10}	1.58E+00	0.00E+00	0.00E+00	3.36E−01	1.94E+02	**0.00E+00**	3.41E+02	5.63E+02	**0.00E+00**
F_{11}	2.54E+03	**0.00E+00**	4.98E−08	8.58E−04	1.76E+01	8.68E−01	1.89E+03	9.22E+03	**0.00E+00**
F_{12}	9.99E+02	**0.00E+00**	**0.00E+00**	2.90E−12	**0.00E+00**	4.46E+00	1.27E+03	2.69E+03	**0.00E+00**
F_{13}	1.23E+03	1.69E+02	6.67E+02	7.19E+02	3.86E+03	**9.82E+01**	2.40E+03	3.17E+03	9.87E+02
F_{14}	3.37E+03	**0.00E+00**	4.03E−01	7.72E−03	1.59E+02	3.46E−01	2.65E+03	1.28E+04	**0.00E+00**
F_{15}	1.93E−12	**0.00E+00**	**0.00E+00**	8.40E−02	1.84E+01	**0.00E+00**	2.11E+01	3.45E+418	**0.00E+00**
F_{16}	8.02E+03	**0.00E+00**	**0.00E+00**	1.67E−12	0.00E+00	3.92E+00	2.23E+03	5.38E+03	**0.00E+00**
F_{17}	3.55E+11	6.71E+01	1.71E+02	2.18E+02	1.98E+02	**7.83E+00**	2.80E+03	7.60E+03	1.20E+01
F_{18}	2.03E+03	**0.00E+00**	3.28E−12	1.31E−07	8.43E+00	5.89E−01	9.12E+02	5.68E+03	**0.00E+00**
F_{19}	2.05E+03	**0.00E+00**	**0.00E+00**	2.10E−01	1.12E+02	**0.00E+00**	2.57E+02	9.65E+04	**0.00E+00**

Table 12. Comparison of mDE-bcM and the second group of algorithms for 25 runs on functions F_1–F_{19}, where $D = 1\,000$

	MOS-CEC2013	MOS-SOCO2011	jDElscop	GaDE	DECC-G	CCJADE	L-SHADE	LM-CMA-ES	mDE-bcM
Ranking	4.11	1.68	2.37	3.00	4.63	3.16	5.53	6.05	1.68
# Best	4	14	8	2	4	6	0	0	15
nWins-better	14	3	9	15	15	12	18	18	–
nWins-similar	3	13	8	2	3	4	0	0	–
nWins-worse	2	3	2	2	1	3	1	1	–
p-value	2.70E−03	1	3.48E−02	1.60E−04	3.00E−03	2.01E−02	1.31E−05	9.62E−05	–

Table 13. Average ranking, number of functions with the best results, nWins value and the Friedman test for mDE-bcM and the second group of algorithms for 25 runs on functions $F_1 - F_{19}$, where $D = 1\,000$

The Friedman test reported a p-value = 6.51E−14 for Table 12, which is below the significance level $\alpha = 0.05$, and means that there are statistical differences as the negation of the null hypothesis. Used mDE-bcM as the control algorithm, further the Friedman tests show that mDE-bcM has been significant differences with MOS-CEC2013, GaDE, CCJADE, L-SHADE and LM-CMA-ES, for jDElscop and DECC-G, the p-values are nonetheless very close to the significance level, but for MOS-SOCO2011, the p-value = 1 shows that there are not significant differences between mDE-bcM and MOS-SOCO2011, MOS-SOCO2011 is indeed a competitive algorithm.

It is worth noting that MOS-CEC2013 performs very well on the CEC 2013 benchmark suite, but generally on the SOCO 2011 benchmark suite, which proves once again the limitation of the algorithm in solving the problem.

7 CONCLUSIONS

This research proposes a new algorithm called mDE-bcM for solving large-scale global optimization problems. The mDE-bcM divides the population into three subpopulations with the fitness values, the second and the third of which employs

a modified mutation strategy called best-and-current mutation strategy, three subpopulations evolved independently and then fused after one evolution. The mDE-bcM was tested on a set of benchmark functions provided for the Soft Computing special issue on scalability of evolutionary algorithms for large-scale continuous optimization problems. After comparing with other 16 state-of-the-art algorithms in use, it shows a very competitive performance on the SOCO 2011 benchmark suite.

For future work, we intend to use ensemble mutation strategies and success rate of mutation strategies for difficult problems, and test our algorithm on extra sets of competitive LSGO benchmarks, such as presented in the CEC'11 [47] and CEC'13 [32] special sessions.

Acknowledgement

This work is supported by the NSFC (National Natural Science Foundation of China) project (grants No. 62066041, 41861047) and the Northwest Normal University young teachers' scientific research capability upgrading program (NWNU-LKQN-17-6).

REFERENCES

[1] Storn, R.—Price, K.: Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization, Vol. 11, 1997, No. 4, pp. 341–359, doi: 10.1023/A:1008202821328.

[2] Price, K.—Storn, R. M.—Lampinen, J. A.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Berlin, Heidelberg, Natural Computing Series, 2005, doi: 10.1007/3-540-31306-0.

[3] Das, S.—Suganthan, P. N.: Differential Evolution: A Survey of the State-of-the-Art. IEEE Transactions on Evolutionary Computation, Vol. 15, 2011, No. 1, pp. 4–31, doi: 10.1109/TEVC.2010.2059031.

[4] Das, S.—Mullick, S. S.—Suganthan, P. N.: Recent Advances in Differential Evolution – An Updated Survey. Swarm and Evolutionary Computation, Vol. 27, 2016, pp. 1–30, doi: 10.1016/j.swevo.2016.01.004.

[5] Floudas, C. A.—Gounaris, C. E.: A Review of Recent Advances in Global Optimization. Journal of Global Optimization, Vol. 45, 2009, No. 1, pp. 3–38, doi: 10.1007/s10898-008-9332-8.

[6] Maučec, M. S.—Brest, J.: A Review of the Recent Use of Differential Evolution for Large-Scale Global Optimization: An Analysis of Selected Algorithms on the CEC 2013 LSGO Benchmark Suite. Swarm and Evolutionary Computation, Vol. 50, 2019, Art. No. 100428, doi: 10.1016/j.swevo.2018.08.005.

[7] Qin, A. K.—Huang, V. L.—Suganthan, P. N.: Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. IEEE Trans-

actions on Evolutionary Computation, Vol. 13, 2009, No. 2, pp. 398–417, doi: 10.1109/TEVC.2008.927706.

[8] Mallipeddi, R.—Lee, M.: An Evolving Surrogate Model-Based Differential Evolution Algorithm. Applied Soft Computing, Vol. 34, 2015, pp. 770–787, doi: 10.1016/j.asoc.2015.06.010.

[9] Neri, F.—Tirronen, V.: Recent Advances in Differential Evolution: A Survey and Experimental Analysis. Artificial Intelligence Review, Vol. 33, 2010, No. 1-2, pp. 61–106, doi: 10.1007/s10462-009-9137-2.

[10] Weber, M.—Neri, F.—Tirronen, V.: Distributed Differential Evolution with Explorative-Exploitative Population Families. Genetic Programming and Evolvable Machines, Vol. 10, 2009, No. 4, pp. 343–371, doi: 10.1007/s10710-009-9089-y.

[11] Ali, M. Z.—Awad, N. H.—Suganthan, P. N.: Multi-Population Differential Evolution with Balanced Ensemble of Mutation Strategies for Large-Scale Global Optimization. Applied Soft Computing, Vol. 33, 2015, pp. 304–327, doi: 10.1016/j.asoc.2015.04.019.

[12] Lampinen, J.: Differential Evolution – New Naturally Parallel Approach for Engineering Design Optimization. In: Topping, B. H. V. (Ed.): Developments in Computational Mechanics with High Performance Computing. Civil-Comp Press, Ithaca, UK, 1999, pp. 217–228.

[13] Zaharie, D.: Control of Population Diversity and Adaptation in Differential Evolution Algorithms. In: Matousek, D., Osmera, P. (Eds.): Proceedings of MENDEL International Conference on Software Computing, 2003, pp. 41–46.

[14] Weber, M.—Neri, F.—Tirronen, V.: Shuffle or Update Parallel Differential Evolution for Large-Scale Optimization. Soft Computing, Vol. 15, 2011, No. 11, pp. 2089–2107, doi: 10.1007/s00500-010-0640-9.

[15] Chen, Y.—Lin, Y.—Hu, X.: Parallel Differential Evolution with Multi-Population and Multi-Strategy. Journal of Frontiers of Computer Science and Technology, Vol. 8, 2014, No. 12, pp. 1502–1510.

[16] Wu, G.—Mallipeddi, R.—Suganthan, P. N.—Wang, R.—Chen, H.: Differential Evolution with Multi-Population Based Ensemble of Mutation Strategies. Information Sciences, Vol. 329, 2016, pp. 329–345, doi: 10.1016/j.ins.2015.09.009.

[17] Zhang, J.—Sanderson, A. C.: JADE: Adaptive Differential Evolution with Optional External Archive. IEEE Transactions on Evolutionary Computation, Vol. 13, 2009, No. 5, pp. 945–958, doi: 10.1109/TEVC.2009.2014613.

[18] Kong, X.—Gao, L.—Ouyang, H.—Shao, Y.: Adaptive Differential Evolution Algorithm with Bidirectional Randomly Multi-Mutation Strategy. Computer Integrated Manufacturing Systems, Vol. 20, 2014, No. 8, pp. 1948–1958, doi: 10.13196/j.cims.2014.08.kongxiangyong.1948.11.20140817 (in Chinese).

[19] Zhang, X.—Yuen, S. Y.: A Directional Mutation Operator for Differential Evolution Algorithms. Applied Soft Computing, Vol. 30, 2015, pp. 529–548, doi: 10.1016/j.asoc.2015.02.005.

[20] Qiu, X.—Jiang, Y.—Li, B.: Fractal Mutation Factor Correcting Differential Evolution Algorithm. Pattern Recognition and Artificial Intelligence, Vol. 28, 2015, No. 2, pp. 132–138, doi: 10.16451/j.cnki.issn1003-6059.201502005 (in Chinese).

[21] Parouha, R. P.—Das, K. N.: A Memory Based Differential Evolution Algorithm for Unconstrained Optimization. Applied Soft Computing, Vol. 38, 2016, pp. 501–517, doi: 10.1016/j.asoc.2015.10.022.

[22] Elsayed, S. M.—Sarker, R. A.—Essam, D. L.: Differential Evolution with Multiple Strategies for Solving CEC2011 Real-World Numerical Optimization Problems. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2011), New Orleans, LA, USA, IEEE, 2011, pp. 1041–1048, doi: 10.1109/CEC.2011.5949732.

[23] Tong, L.—Dong, M.—Jing, C.: An Improved Multi-Population Ensemble Differential Evolution. Neurocomputing, Vol. 290, 2018, pp. 130–147, doi: 10.1016/j.neucom.2018.02.038.

[24] Xu, B.—Tao, L.—Chen, X.—Cheng, W.: Adaptive Differential Evolution with Multi-Population-Based Mutation Operators for Constrained Optimization. Soft Computing, Vol. 23, 2019, pp. 3423–3447, doi: 10.1007/s00500-017-3001-0.

[25] Herrera, F.—Lozano, M.—Molina, D.: Test Suite for the Special Issue of Soft Computing on Scalability of Evolutionary Algorithms and Other Metaheuristics for Large Scale Continuous Optimization Problems. Technical Report, SCI2S, University of Granada, Spain, 2010.

[26] Lozano, M.—Molina, D.—Herrera, F.: Editorial Scalability of Evolutionary Algorithms and Other Metaheuristics for Large-Scale Continuous Optimization Problems. Soft Computing, Vol. 15, 2011, No. 11, pp. 2085–2087, doi: 10.1007/s00500-010-0639-2.

[27] Zielinski, K.—Peters, D.—Laur, R.: Run Time Analysis Regarding Stopping Criteria for Differential Evolution and Particle Swarm Optimization. Proceedings of the 1st International Conference on Experiments/Process/System Modelling/Simulation/Optimization, 2005, pp. 1–8.

[28] Opara, K.—Arabas, J.: Differential Evolution: A Survey of Theoretical Analyses. Swarm and Evolutionary Computation, Vol. 44, 2019, pp. 546–558, doi: 10.1016/j.swevo.2018.06.010.

[29] LaTorre, A.—Muelas, S.—Peña, J.-M.: A Comprehensive Comparison of Large Scale Global Optimizers. Information Sciences, Vol. 316, 2015, pp. 517–549, doi: 10.1016/j.ins.2014.09.031.

[30] LaTorre, A.—Muelas, S.—Peña, J.-M.: Large Scale Global Optimization: Experimental Results with MOS-Based Hybrid Algorithms. 2013 IEEE Congress on Evolutionary Computation (CEC 2013), Cancún, Mexico, 2013, pp. 2742–2749, doi: 10.1109/CEC.2013.6557901.

[31] Tang, K.—Li, X.—Suganthan, P. N.—Yang, Z.—Weise, T.: Benchmark Functions for the CEC 2010 Special Session and Competition on Large Scale Global Optimization. Technical Report, 2010.

[32] Li, X.—Tang, K.—Omidvar, M. N.—Yang, Z.—Qin, K.: Benchmark Functions for the CEC 2013 Special Session and Competition on Large Scale Global Optimization. Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.

[33] LaTorre, A.—Muelas, S.—Peña, J.: A MOS-Based Dynamic Memetic Differential Evolution Algorithm for Continuous Optimization: A Scalability Test. Soft Computing, Vol. 15, 2011, No. 11, pp. 2187–2199, doi: 10.1007/s00500-010-0646-3.

[34] Falco, I.—Cioppa, A.—Trunfio, G.: Investigating Surrogate-Assisted Cooperative Coevolution for Large-Scale Global Optimization. Information Sciences, Vol. 482, 2019, pp. 1–26, doi: 10.1016/j.ins.2019.01.009.

[35] Derrac, J.—García, S.—Molina, D.—Herrera, F.: A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms. Swarm and Evolutionary Computation, Vol. 1, 2011, No. 1, pp. 3–18, doi: 10.1016/j.swevo.2011.02.002.

[36] Eshelman, L. J.—Schaffer, J. D.: Real-Coded Genetic Algorithms and Interval-Schemata. Foundations of Genetic Algorithms, Vol. 2, 1993, pp. 187–202, doi: 10.1016/B978-0-08-094832-4.50018-0.

[37] Auger, A.—Hansen, N.: A Restart CMA Evolution Strategy with Increasing Population Size. 2005 IEEE Congress on Evolutionary Computation, Edinburgh, Scotland, UK, 2005, Vol. 2, pp. 1769–1776, doi: 10.1109/CEC.2005.1554902.

[38] Molina, D.—Lozano, M.—Sanchez, A. M.—Herrera, F.: Memetic Algorithms Based on Local Search Chains for Large Scale Continuous Optimisation Problems: MA-SSW-Chains. Soft Computing, Vol. 15, 2011, No. 11, pp. 2201–2220, doi: 10.1007/s00500-010-0647-2.

[39] Tseng, L.—Chen, C.: Multiple Trajectory Search for Large Scale Global Optimization. 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Hong Kong, China, 2008, pp. 3052–3059, doi: 10.1109/CEC.2008.4631210.

[40] Liang, J. J.—Suganthan, P. N.—Deb, K.: Novel Composition Test Functions for Numerical Global Optimization. Proceedings 2005 IEEE Swarm Intelligence Symposium (SIS 2005), 2005, pp. 68–75, doi: 10.1109/SIS.2005.1501604.

[41] Duarte, A.—Marti, R.—Gortazar, F.: Path Relinking for Large-Scale Global Optimization. Soft Computing, Vol. 15, 2011, No. 11, pp. 2257–2273, doi: 10.1007/s00500-010-0650-7.

[42] Brest, J.—Maučec, M. S.: Self-Adaptive Differential Evolution Algorithm Using Population Size Reduction and Three Strategies. Soft Computing, Vol. 15, 2011, No. 11, pp. 2157–2174, doi: 10.1007/s00500-010-0644-5.

[43] Yang, Z.—Tang, K.—Yao, X.: Scalability of Generalized Adaptive Differential Evolution for Large-Scale Continuous Optimization. Soft Computing, Vol. 15, 2011, No. 11, pp. 2141–2155, doi: 10.1007/s00500-010-0643-6.

[44] Yang, Z.—Tang, K.—Yao, X.: Large Scale Evolutionary Optimization Using Cooperative Coevolution. Information Sciences, Vol. 178, 2008, No. 15, pp. 2985–2999, doi: 10.1016/j.ins.2008.02.017.

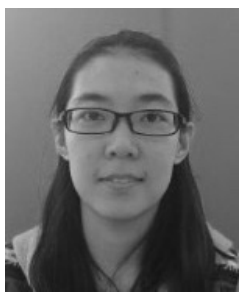
[45] Tanabe, R.—Fukunaga, A. S.: Improving the Search Performance of SHADE Using Linear Population Size Reduction. 2014 IEEE Congress on Evolutionary Computation (CEC 2014), 2014, pp. 1658–1665, doi: 10.1109/CEC.2014.6900380.

[46] Loshchilov, I.: A Computationally Efficient Limited Memory CMA-ES for Large Scale Optimization. Proceedings of the 2014 Annual Conference on Ge-

netic and Evolutionary Computation (GECCO '14), 2014, pp. 397–404, doi: 10.1145/2576768.2598294.

[47] Li, X.—Tang, K.—Omidvar, M.—Yang, Z.—Qin, K.: Benchmark Functions for the CEC '2011 Special Session and Competition on Large Scale Global Optimization. Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.

Yongjie Ma graduated from the University of Electronic Science and Technology of China with a Master's degree in electromagnetic measurement technology and instruments in 1998. He received his Ph.D. in traffic information and control from Lanzhou Jiaotong University in 2011. He is currently Professor in the School of Physics and Electronic Engineering of Northwest Normal University. His research interests include evolutionary algorithms and their engineering application, intelligent transportation image processing. He published more than 70 technical papers in journals.



Lin Zhu graduated from Northwest Normal University in 2017 with a Master's degree. Her main research interest are evolutionary algorithms, and she has published 3 research papers in journals.

Yulong Bai received his Ph.D. in natural science (cartography and geography information systems) from Graduate University of Chinese Academy of Science, China in January 2012. He is currently Professor at College of Physics and Electrical Engineering, Northwest Normal University, China. His research interests include control theory and application, data assimilation methods and chaos. He published more than 30 papers in peer-reviewed journals.

Computing and Informatics, Vol. 39, 2020, 510–536, doi: 10.31577/cai_2020_3_510

INVESTIGATION OF PARALLEL DATA PROCESSING USING HYBRID HIGH PERFORMANCE CPU + GPU SYSTEMS AND CUDA STREAMS

Paweł Czarnul

Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology, Narutowicza 11/12, 80-233
Gdansk, Poland
e-mail: pczarnul@eti.pg.edu.pl

Abstract. The paper investigates parallel data processing in a hybrid CPU + GPU(s) system using multiple CUDA streams for overlapping communication and computations. This is crucial for efficient processing of data, in particular incoming data stream processing that would naturally be forwarded using multiple CUDA streams to GPUs. Performance is evaluated for various compute time to host-device communication time ratios, numbers of CUDA streams, for various numbers of threads managing computations on GPUs. Tests also reveal benefits of using CUDA MPS for overlapping communication and computations when using multiple processes. Furthermore, using standard memory allocation on a GPU and Unified Memory versions are compared, the latter including programmer added prefetching. Performance of a hybrid CPU+GPU version as well as scaling across multiple GPUs are demonstrated showing good speed-ups of the approach. Finally, the performance per power consumption of selected configurations are presented for various numbers of streams and various relative performances of GPUs and CPUs.

Keywords: GPGPU, overlapping computations and communication, MPS, Unified Memory, performance, power consumption

Mathematics Subject Classification 2010: 68M20, 65Y05, 68N15

1 INTRODUCTION

In today's high performance computing (HPC) systems, several computing devices are typically used – multi- and many-core CPUs, GPUs, FPGAs. All have their advantages and disadvantages depending on particular types of codes and applications [9]. Most of HPC systems nowadays feature either traditional multicore CPU + accelerator (GPU, Intel Xeon Phi x100) or manycore CPUs (such as Intel Xeon Phi x200 or Sunway manycore CPUs in the Sunway TaihuLight cluster). Selected application examples of applications running on such systems include data encryption and decryption algorithms [32], pattern matching for deep packet inspection [26], RNA secondary structure prediction [27], parallel implementation for a DVB-RCS2 receiver [46], parallelization of large vector similarity computations [14, 11], stitching large scale optical microscopy images [4], etc. For this reason, efficient management of computations among these processors is a key to achieving high throughput, especially for incoming data streams that must be processed under time constraints. GPGPU has become very popular for processing large data sets in the Single Instruction Multiple Threads fashion. As long as processing in threads does not result in too much divergence, one can achieve very high processing throughput. Especially important is also fast delivering of input data from host memory to GPU memory and results back from GPU memory to the host. This can be achieved through overlapping communication and GPU and CPU computations by using multiple streams. This topic is investigated in this paper in detail, in terms of performance for various numbers of streams, threads managing computations in a CPU + GPU setting, using the standard GPU memory management and Unified Memory [34] approaches. Furthermore, as today's HPC is not only about performance, power consumption is also considered, in the context of performance to power consumption ratio of various configurations.

The approach adopted in this paper includes analysis based on a custom built benchmark, described in Section 3, that assumes input data that is composed of multiple data chunks which are fed into CUDA streams to GPUs or processed on multicore host CPUs. The benchmark allows for various compute time to host-device communication time ratios, numbers of streams and threads managing computations and communication and thus, depending on the values of parameters, can be regarded as a template representative of many real world applications.

The objective of this work is to assess performance and selected performance/power characteristics of parallel processing of a data stream which is passed for computations to either GPUs using CUDA streams or to GPUs and CPU cores in a hybrid CPU + GPU approach. The contribution includes assessment of preferred numbers of streams for various GPU architectures, preferred application architecture in terms of the number of host GPU management and computing threads, assessment of performance differences between standard memory management, Unified Memory and Unified Memory with prefetching, all for various compute to communication ratios. Additionally, performance per power consumption is evaluated for selected configurations. Furthermore, scaling from 1 to 4 NVIDIA Tesla V100 GPUs

of DGX Station installed at the Faculty of ETI, Gdansk University of Technology, is presented.

The outline of the paper is as follows. Section 2 presents the existing related work and contributions of this paper in that context, Section 3 the processing model and design of the benchmark used for experiments, Section 4 tests and results including testbed systems, impact of multiple streams on performance using various numbers of threads managing computations, launching computations from multiple processes with and without MPS, performance with and without Unified Memory, scalability of hybrid CPU + GPU code, scaling across multiple GPUs and performance-power consumption ratios for hybrid configurations. Finally Section 5 presents conclusions and future work.

2 RELATED WORK

2.1 Mechanisms for Data Management in Selected GPU-Aware Parallel Programming APIs

Overlapping computation on the GPU, CPU as well as CPU-GPU and GPU-CPU communication is a well known technique that allows to minimize execution time of an application using GPUs [14, 31, 13, 25]. This approach can be used for both batch processing if the data is already available when the application starts or is incoming to a node in possibly many data streams.

In CUDA, kernel functions are executed in parallel on a GPU by a grid which is composed of thread blocks each of which consists of a number of threads. Blocks within a grid and threads within a block can be lined up in 1, 2 or 3 dimensions. Various operations (out of host-to-device communication, device-to-host communication, kernel execution) submitted to two different streams can potentially be overlapped in H2D, compute and D2H queues. Thus, a larger number of streams can potentially allow better overlapping (so-called n-way in the case of n streams [41]) if there is potential for that in the application and if the GPU and the driver support that. Potentially kernels can also be executed in parallel, depending on their requirements and the GPU. Unified Memory allows allocation and access to data from the host and device sides and page migration, transparent to the user. The contribution of the paper is how a particular configuration (with a given number of streams) for a given GPU (GPUs of various architectures were used) benefits which is otherwise very difficult to predict given these factors.

It should be noted that OpenCL offers a similar programming model to CUDA but targeting systems with both GPUs as well as CPUs [13, 15]. Specifically, a kernel can be executed on a compute device by a structure called NDRange that consists of work groups which in turn consist of work items. Both work groups within the NDRange and work items within a work group can be lined up in 1, 2 or 3 dimensions. A kernel is executed by work items in parallel within a context that is associated with one or more devices. Input and output data are managed through memory objects. Overlapping can be achieved using command queues, similarly to

using streams in CUDA. OpenCL version 2.0+ allows to use Shared Virtual Memory which allows codes running on the host and a device to share data. Various modes including coarse-grained or fine-grained with the possibility of accessing locations concurrently if SVM atomic operations are supported. Another high level API allowing to use GPUs in a way similar to OpenMP is OpenACC [13, 15]. OpenACC allows to use directives for instructing parallelization of code regions, specifically loops as well as scoping of data and synchronization. Data related directives allow to specify allocation, releasing memory and rely on the concept of reference counters to data.

Assessment of benefits and the performance of Unified Memory was done previously in [22], but it was only for batch type input data for applications such as verification of Goldbach's conjecture, 2D heat transfer analysis and adaptive numerical integration. That research was then extended with evaluation of not only the basic Unified Memory code against the standard approach but also Unified Memory with prefetching [23]. Results were presented for four applications: Sobel and image rotation filters as well as stream image processing and computational fluid dynamic simulation. Tests were performed on Pascal and Volta architecture GPUs, specifically NVIDIA GTX 1080 and NVIDIA V100 cards. Furthermore, evaluation of Unified Memory oversubscription over the standard manual management approach was provided, generally showing slight benefits of the latter, if implemented efficiently. In those contexts, the contribution of this paper is assessment of impact of the number of streams with Unified Memory, assessment of NVIDIA MPS's performance and consideration of power consumption with the number of streams in parallel processing with CUDA.

2.2 Selected Works on Efficiency of Using Multiple Streams Using GPUs

There are studies in the literature on efficiency of using multiple streams using GPUs. For instance, paper [20] investigates the impact of using various numbers of streams on the performance of such an application with a theoretical formula for the best number of streams. It was considered in terms of the number of iterations of a loop within a kernel. Tests were performed for GTX 280 and GTX 480 cards which are not widely used anymore. GPU architectures have also changed considerably since then. In paper [12], the author analyzed and compared the performance of processing on a GPU using 1, 2 and 4 streams for modern GPUs: mobile NVIDIA GeForce 940MX, desktop GTX 1060, server Tesla K20m and Tesla V100. Tests were performed for various compute time to host-device communication time ratios proving large benefits of using 2 or 4 streams for overlapping communication and computations and showing relative performances of the tested GPUs. Compared to [20] this paper contributes by analysis on newer GPUs, consideration of Unified Memory approach and performance to power consumption analysis. Compared to [12] this paper brings testing using more streams, multi-threaded and single-threaded applications, MPS as well as performance to power consumption

considerations. Apart from multiple streams, concurrent kernel execution is also possible on GPUs. Paper [45] investigates approaches such as context switching, manual context funneling and automatic CUDA context funneling but tests were performed on older CUDA 4 and earlier versions and demonstrated that automatic CUDA context funneling (sharing a context among process threads) is very efficient. Work [29] proposes a detailed computation-bound single kernel performance model for understanding the resource scheduling system with CUDA streams and focuses on multi-kernel concurrency. Similarly, paper [8] investigates conditions needed for concurrent execution of kernels simultaneously.

2.3 Using Multiple Streams for Various Applications

Deployment of multi-stream processing for GPU based systems for particular applications has been analyzed in the literature. In paper [43] authors focus on performance improvement through more effective overlapping of communication and computations using OpenMP as well as multiple CUDA threads. Many threads control each GPU and the authors have launched 4 CUDA streams for each pair of neighboring GPUs to overlap communication and computation of inner domain points. A 3D stencil use case was used to demonstrate benefits over previous solutions. Tests were performed on Kepler and Fermi cards. Compared to that work, this paper considers a model with independent input data chunks rather than geometric Single Program Multiple Data paradigm [13], considers more streams, Unified Memory and power consumption for a more recent Pascal card. In paper [19] authors focus on improvement of performance of Sparse matrix-vector multiplication (SpMV) code using many GPUs installed within a node. Optimization is performed using multiple OpenMP threads that control particular GPUs as well as multiple CUDA streams for overlapping. Benefits of such improved approach using 2 GPUs are shown against a naive 1 GPU system implementation for a variety of sparse matrices. Compared to that approach, this paper considers hybrid CPU + GPU processing, investigates multiple streams, Unified Memory and performance to power consumption ratios. Paper [35] proposes a multi-stream implementation of stereo disparity estimation and anaglyph video frame generation using GPUs. Specifically, multiple threads are started using Pthreads, each of which manages a certain number of streams. Performance is presented for a thread count between 1 and 8 and the number of streams between 1 and 8 showing considerable speed-ups of the solution with 100 frames per second for $1\,024 \times 1\,024$ color images. GeForce GTX780 cards where used for experiments. Paper [38] contributes by proposal of a parallel CPU + GPU code for image formation in scanning transmission electron microscopy. Similarly to this work, an algorithm for parallelization using multicore CPUs and GPUs are provided, with assessment of benefits from using multiple CUDA streams. In that context, this paper contributes by analysis of various numbers of streams, Unified Memory and performance to power consumption ratios for similar computations. Utilization of CUDA streams for parallel implementation of a genetic algorithm is presented in paper [39]. Data stream processing accelerated

using GPUs in the context of DBMSes is discussed in [36] for data representation better matching the GPU architecture. Similarly, this paper contributes by consideration of various stream and thread CPU + GPU configurations, Unified Memory and performance to power consumption ratios.

2.4 Selected Frameworks and Environments for Processing Data Using GPUs

Paper [24] provides analysis of programming environments for processing large amounts of data efficiently. Specifically, the work investigates programmability vs. performance such that programs can increase their performance at the cost of decreasing programmability. Java and Stream API, C/C++ and OpenMP, C/C++ and CUDA (with and without CUDA streams) are compared. Power-aware computations for data processing is also an important research topic considered today [16]. There exist frameworks that provide higher than OpenMP, CUDA and MPI programming abstractions to processing data streams using GPUs, good performance and relatively easy-to-use programming models. Available solutions for data streaming include, in particular, Spark [47], Storm [28, 21], Storm working in a geographically distributed and highly variable environment [6], FastFlow [2], extension of FastFlow for a network of multi-core workstations [1], Flink [5, 18], PiCo [33], Thrill [3]. Paper [48] describes GStream that is a scalable framework suited for a cluster of GPUs with GStream API over CUDA, Pthreads and MPI. It is demonstrated for benchmarks such as FIR, MM, FFT, IS and LAMMPS that it offers very good speed-ups, only slightly worse that raw CUDA. For this and the following high level approaches, the contributions of this paper can be used for improvement of performance of lower level building blocks and mapping computations onto GPUs and CPUs as well as optimization of CPU-GPU communication. Another general data processing platform utilizing GPUs is G-Storm [7] which can be used for various applications and data types and provides a high level programming approach. It handles data transfers and resource allocation automatically. If data is to be further used on the same GPU in subsequent operations, it will not be copied back and forth between the host and the GPU. G-Storm very much relies on CUDA MPS that allows to create a single context that can be used from many processes on the host. It should be noted that this paper evaluates gains from MPS and shows benefits of multi-threaded and CUDA multi-stream approach for even better performance and such can be used to improve existing systems. Paper [40] proposes an efficient real-time system for processing large amounts of high frequency data such as video and text. The approach integrates Hadoop for parallel processing, Spark for the real-time component and GPUs for processing. Matrix type data is processed on GPUs similarly to MapReduce. The authors conclude that the proposed solution is faster than CPU MapReduce. Such a system could also benefit from low level optimization between host and GPUs presented in this paper. Work [44] proposes a CPU + GPU system for processing a large number of incoming data streams with hard real-time constraints. A scheduler running on the CPU side distributes

streams among CPUs and GPUs for high utilization of the system in order to meet the constraints. The solution was evaluated using an AES-CBC encryption kernel on thousands of streams proving over 80 % more data processing rate than a single GPU system. Paper [42] presents KernelHive that can be used to optimize scheduling and execution of processing using a stream of multiple independent data chunks on hybrid CPU + GPU systems. Efficient multithreaded data stream processing in a workflow management system called BeesyCluster, either within a high performance workstation or even spanning multiple clusters, is presented in paper [10]. In that context, the contribution of this paper is optimization of internal building blocks for efficient GPU management and consideration of power consumption as well.

3 PROCESSING MODEL AND DESIGN OF BENCHMARK

This section presents the custom-developed application benchmark that is representative of various applications run on GPUs or in a hybrid CPU + GPU environment. Many variables have been considered and can be changed in the proposed processing model and as such were used for subsequent tests. Design of the benchmark application is shown in Figure 1. It is assumed that the application processes a sequence of input data packets such that two data packets serve as input to a processing function that produces output data. This general assumption corresponds to many real life applications, depending on relative sizes of output and input data, e.g. multiplication, addition or other operations on matrices that are important computational steps in various artificial intelligence applications such as deep neural network training. Parallelization involves the following elements and ideas:

1. At a high level of parallelism, OpenMP threads are spawned – one thread per each GPU and additionally one thread managing computations on a multi-core CPU(s). These threads fetch input data from memory in a critical section and pass for computations either to a GPU or the CPU(s). This scheme, working in a loop, effectively supports dynamic load balancing among compute devices.
2. Nested OpenMP parallelism is used for parallelization with many threads on the CPU(s).
3. Input data can be stored in regular RAM from which it can be sent to GPU's global memory explicitly or stored in previously allocated space in Unified Memory. In the latter case, prefetching can be turned on for enabling overlapping computations with host-device communication. In the case of the Unified Memory based version, streams are still used for maximum concurrency of operations [34].

The benchmark allows to set various modes and parameters and correspondingly allows to mimic behavior of various applications following the assumed processing pattern:

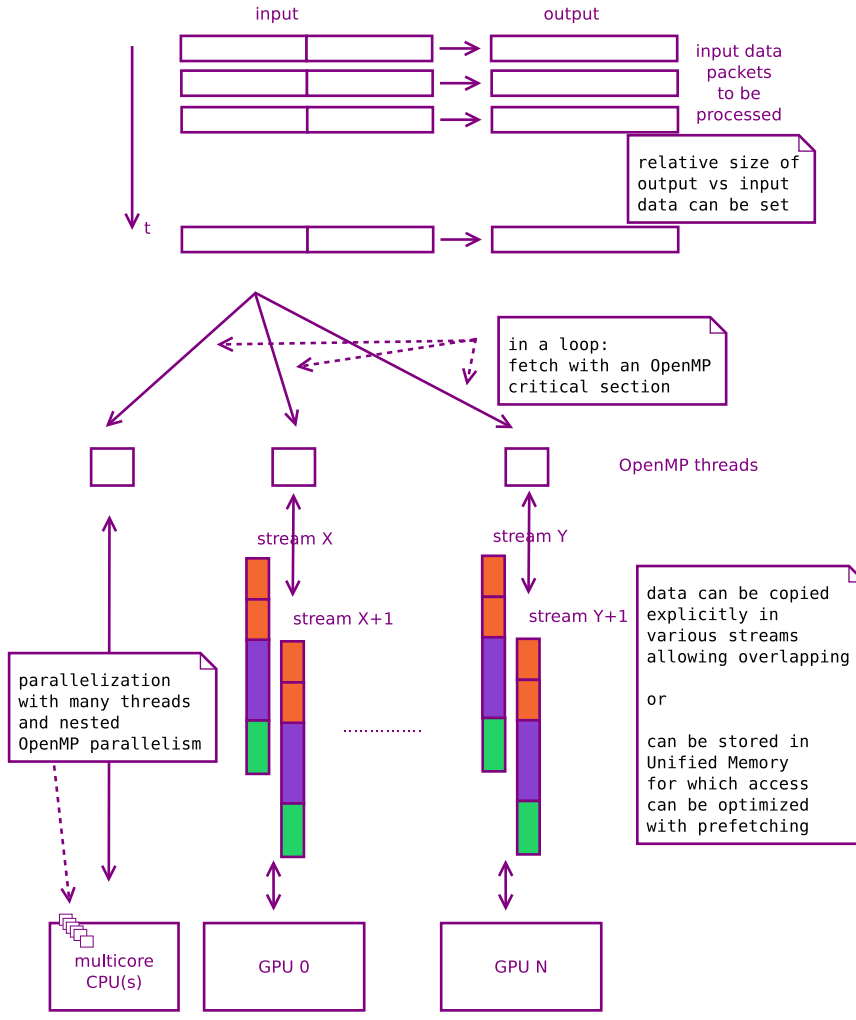


Figure 1. Proposed processing framework

1. memory mode – several modes are possible:

   (a) allocation of host memory *std* using `cudaHostAlloc()` with flag `cudaHostAllocPortable` that does allow subsequent overlapping computations and communication in various streams,

   (b) allocation of memory *UM* using Unified Memory (UM) by calling `cudaMallocManaged()` that allows to use the same pointer from both host threads to write input data, from a kernel to read input data and write output results as well as from the host to read output.

   (c) allocation of memory *UMprefetch* using Unified Memory with data prefetching through `cudaMemPrefetchAsync()` for streams to be used in subsequent steps,

2. compute time to host-device communication time ratio that corresponds to the computational time on a given input data chunk divided by the communication time of this data chunk (CPU-GPU-CPU),
3. output-input ratio that denotes the ratio of the size of output data to the size of input data,
4. stream count – the number of streams per one GPU used,
5. host thread count – the number of threads among which computations are scheduled on CPU(s) cores,
6. GPU count ⟨ids of GPUs⟩ – the number and ids of GPU(s) to be used for computations.

In each experiment, unless otherwise noted, data chunk was 256 KB in size and 1.6 GBs of data was processed. In the test we assumed 1 024 threads per block and the total number of threads was 262 144. In the GPU kernel function, a thread fetches its unique index in a grid and processes data from two input arrays into a result stored in its own location (depending on its index) in an output array. Specifically, it computes averages of selected vector elements of the two input arrays and computes a distance between the averages which is added to the final output. All arrays are stored in global memory and the kernel uses 3 variables as temporary indices and one variable as a loop counter. Compute time to host-device communication time ratio is configured with a proper number of iterations of the aforementioned loop.

4 EXPERIMENTS AND TESTS

4.1 Testbed Systems

For experiments, we used the benchmark described in Section 3 run on three modern multicore CPU(s) + GPUs workstations. Specifications of the systems are listed in Table 1. Testbeds 1 and 2 feature 2 Intel Xeon CPUs + 2 NVIDIA GPUs, of various generations while testbed 3 an Intel Xeon CPU + 4 NVIDIA Tesla V100 cards used for testing scaling across multiple GPUs.

For each particular configuration, unless otherwise noted, 10 tests were performed and the average value is presented.

4.2 Impact of Multiple Streams on Performance

The purpose of the following experiments is to determine the impact of using multiple streams for overlapping computations and communication and finally execution time of a GPU enabled application.

Testbed	1	2	3
CPUs	2 × Intel Xeon CPU E5-2620v4 @ 2.10 GHz	2 × Intel(R) Xeon(R) CPU E5-2640 @ 2.50 GHz	Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20 GHz
CPUs – total number of physical/logical cores	16/32	12/24	20/40
System memory size (RAM) [GB]	128	64	256
GPUs	2 × NVIDIA GTX 1070 (Pascal)	2 × NVIDIA Tesla K20m (Kepler)	4 × NVIDIA Tesla V100 (Volta)
GPUs – total number of CUDA cores	2 × 2 048	2 × 2 496	4 × 5 120
GPU Compute capability	6.1	3.5	7.0
GPU memory size [MB]	2 × 8 192	2 × 5 120	4 × 16 384
Operating system	Ubuntu Linux version 4.15.0-36-generic	CentOS Linux version 3.10.0-862.9.1.el7.x86_64	Ubuntu Linux version 4.4.0-83-generic
Compiler/version	CUDA compilation tools, release 9.1, V9.1.85, gcc 7.3.0	CUDA compilation tools, release 9.1, V9.1.85, gcc 4.8.5	CUDA compilation tools, release 9.0, V9.0.176, gcc 5.4.0

Table 1. Testbed configurations

The following tests have been performed for several values of compute time to host-device communication time ratio, for several GPU cards and for the number of streams between 1 and 32. Additionally, two different ways of launching computations on a GPU are presented and compared:

A: One thread per GPU managing computations through one or more streams. In this case, the thread launches CPU-GPU communication, kernel and GPU-CPU communication asynchronously through streams one after another.

B: As many threads as the number of streams are launched per GPU, each of which launches CPU-GPU communication, kernel, GPU-CPU communication in a separate stream. Threads need to synchronize while fetching new input data packets.

Figure 2 presents the results for these versions for particular numbers of threads and streams used for testbed 1 while Figure 3 does so for testbed 2. It can be seen that, in general, best results were obtained using one dedicated host thread per GPU launching communication and computations to multiple streams with 2 streams for testbed 1. For testbed 2 the same implementation offers best results with 2+ streams with small differences between the number of streams larger than 2–4. At the same

time, we can see very small deviations between runs (10 measured) for testbed 2 (default affinity values are presented). For testbed 1, we can observe larger deviations for configurations with multiple host threads launching operations on the GPU (we present threads/close affinity values). These differences might stem from various operating system settings and a compiler version as the CUDA versions were the same.

4.3 Launching Computations from Multiple Processes Using MPS

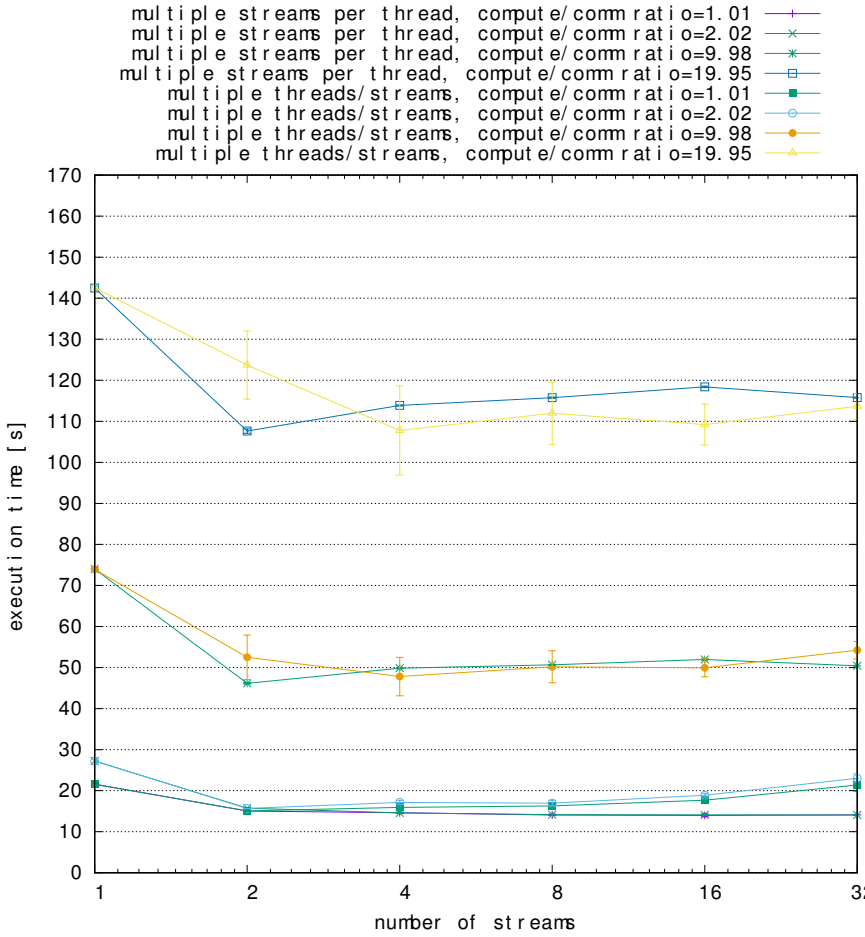


Figure 2. Comparison of implementations with various numbers of threads and streams on a GPU, testbed 1, bars represent standard deviation

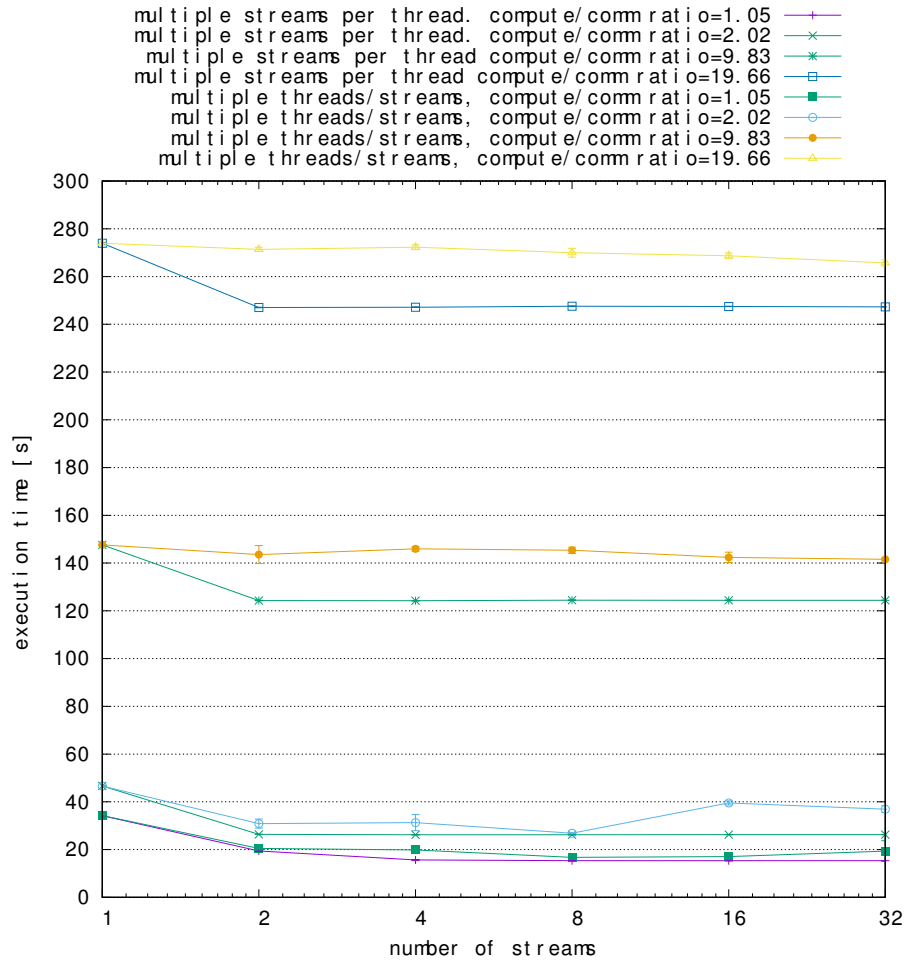


Figure 3. Comparison of implementations with various numbers of threads and streams on a GPU, testbed 2, bars represent standard deviation

In case there is no dedicated parallel application available for parallel processing of incoming data streams to a computer node, it is probable that several processes working in parallel will try to submit the work for processing on a GPU that will be shared in such a case. This may lead to inefficiency of the processing. One solution would involve writing a dedicated multi-stream application as analyzed in this paper. An alternative approach has been made available by NVIDIA through Multi Process Service (MPS) that tries to overlap CPU-GPU communication and processing on a GPU from various contexts. It does not require code modifications which is a considerable advantage. Details of its usage can be found in [13]. Figures 4

and 5 present the results of using the MPS enabled configuration vs the standard configuration for testbed 1 and testbed 2, respectively. Five tests were performed for each configuration and the average value is presented. The results really indicate that the solution improves execution time visibly, except for smaller compute time to host-device communication time ratio for testbed 1. In these tests, two different processes were launched in parallel on the number of data chunks half the sizes of the cases shown in Figures 2 and 3. It should be noted that the best results obtained for 2 streams shown in Figure 2 still offer better execution times while the ones shown in Figure 3 show practically the same or marginally a better performance compared to the one with MPS.

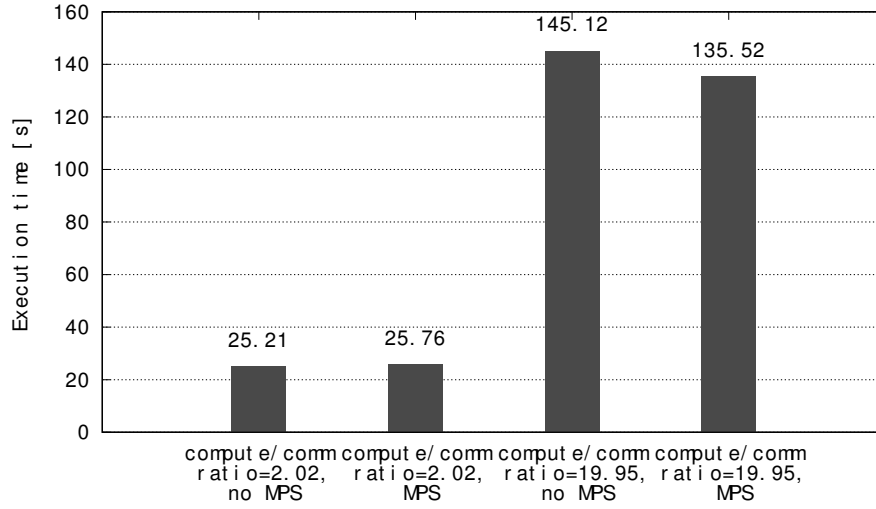


Figure 4. Comparison of performance with and without NVIDIA MPS, testbed 1

4.4 Performance with Unified Memory

Since the latest cards and CUDA versions offer the benefit of easier programming with Unified Memory, this experiment is to show the performance of Unified Memory based implementation compared to previous best cases. The test involves setting input data on the host and launching a kernel that processes data packets on the GPU. Subsequently, results are read from the host side in order to find the maximum of results and display to the user.

The basic UM enabled version was further optimized using data prefetching (we denote this version by UMprefetch). Specifically, the data packet to be processed in a subsequent step in a given stream is prefetched using a call to function `cudaMemPrefetchAsync(...)` on the two input buffers.

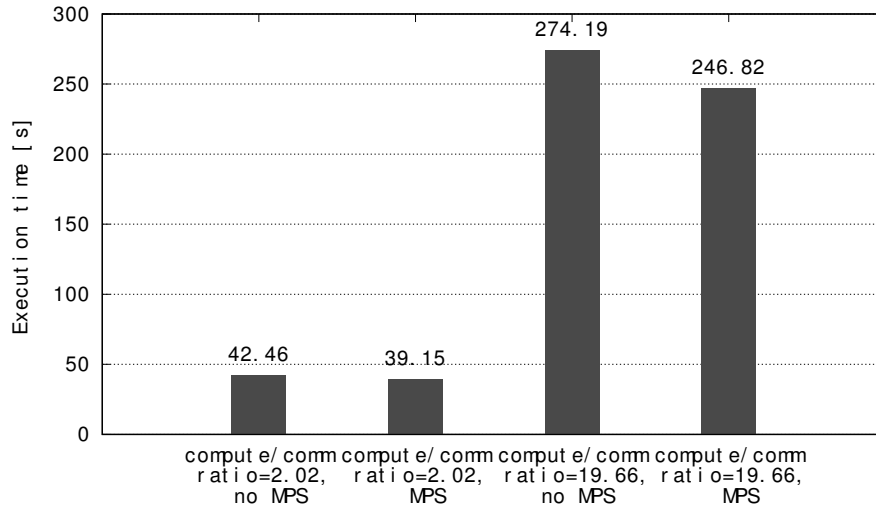


Figure 5. Comparison of performance with and without NVIDIA MPS, testbed 2

Figure 6 presents comparison between std, UM and UMprefetch versions for 1 GPU on testbed 1 while Figure 7 presents comparison between std, UM and UMprefetch versions for 2 GPUs on testbed 1, data size proportionally smaller than in the previous tests. It can be seen that prefetching really improves the performance but still the standard memory optimized multi-stream version offers the best performance. This is in line with some previous works comparing performance of Unified Memory to standard based versions showing generally similar or worse performance in [22], [30] and [37] in return for an easier programming model. This paper confirms it for various compute time to host-device communication time ratios, numbers of streams and 1 and 2 GPUs.

4.5 Scalability of Hybrid CPU + GPU Code

The purpose of the following experiments (using standard memory management) is to show scalability of the hybrid parallel code on the two testbeds with 1 GPU, 2 GPUs as well as host threads engaged for computations, for various GPU/CPU performance ratios. The latter can vary depending on an application. In this case, 2 streams per GPU were used for testbed 1 and 4 streams per GPU for testbed 2. The same thread affinities and binding as in Section 4.2 were used.

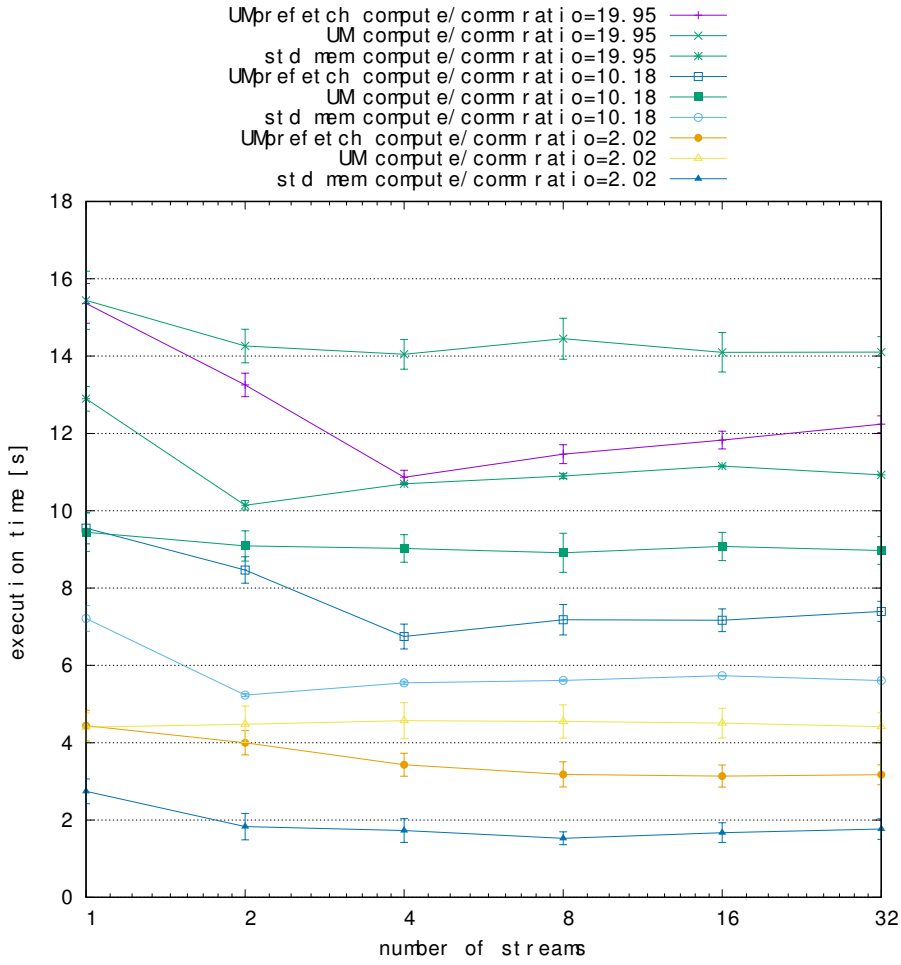


Figure 6. Comparison of standard memory (std), Unified Memory (UM) and optimized Unified Memory (UMprefetch) implementations – testbed 1, 1 GPU, bars represent standard deviation

The results presented in Figure 8 for testbed 1 and in Figure 9 for testbed 2 allow to assess GPU/CPU performances for which adding host threads for computations brings visible savings in execution times. It can be noticed that 2 GPUs configurations achieve relatively better performance than proportional scaling from 1 GPU configurations, apparently due to using one of the GPUs for display as well. Scaling from 1 to 2 GPUs is clearly visible. Increasing the number of host threads decreases application execution time at rates very much depending on GPU to CPU performances, with practically no gains when using 2 GPUs and GPU/CPU per-

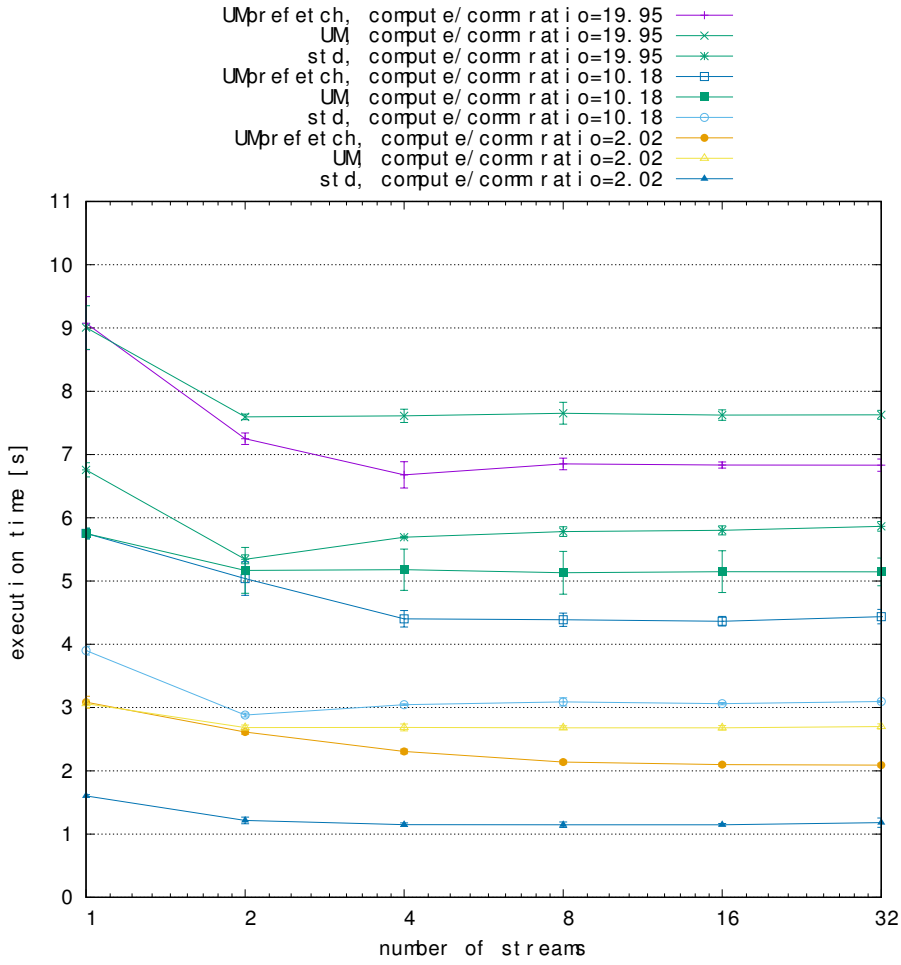


Figure 7. Comparison of standard memory (std), Unified Memory (UM) and optimized Unified Memory (UMprefetch) implementations – testbed 1, 2 GPUs, bars represent standard deviation

formance ratio around 30 for testbed 1. It should be kept in mind that in case some CPU cores are used for computations, still as many threads as the number of GPUs are used for management of computations on the GPUs. Furthermore, the threads managing computations on the GPUs and the CPUs fetch next data packets synchronizing on an OpenMP critical section which also decreases potential speed-ups.

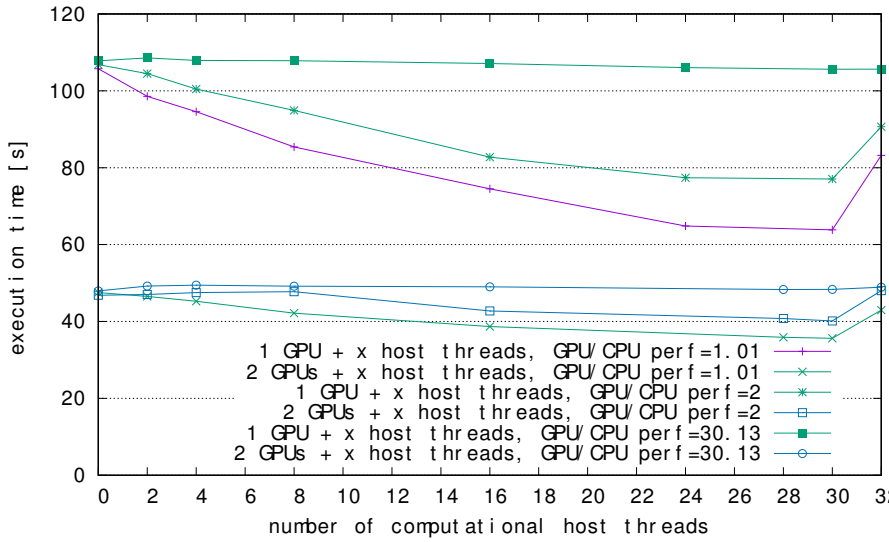


Figure 8. Performance of a hybrid GPU+CPU implementation for various GPU/CPU performances and numbers of host threads, testbed 1

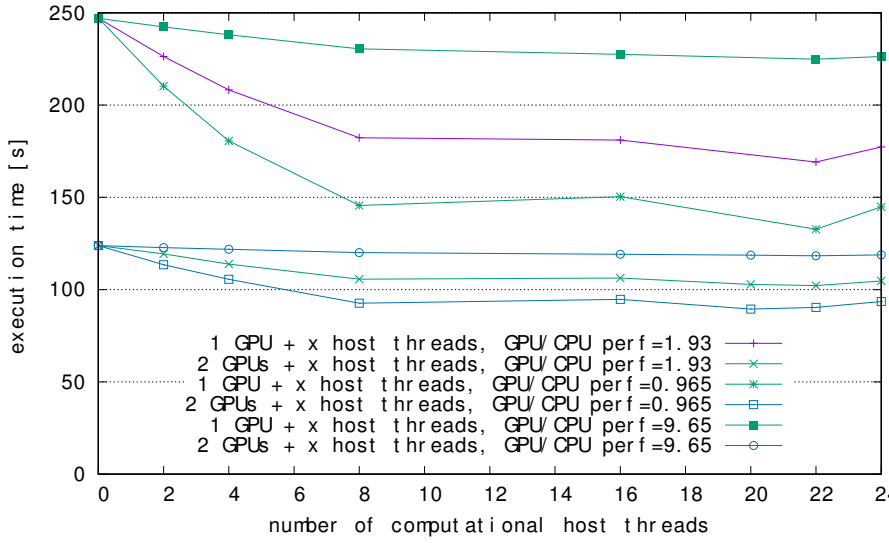


Figure 9. Performance of a hybrid GPU+CPU implementation for various GPU/CPU performances and numbers of host threads, testbed 2

4.6 Performance-Power Consumption Ratio

In today's high performance computing systems, power consumption has become an important topic. It is considered in designs of future clusters for which the total power consumption is suggested not to exceed 20 MW for 1 Exaflop/s [17]. In this context, we analyze the performance to power consumption for the various configurations analyzed in this paper, specifically for:

1. various numbers of streams involved when using 1 GPU,
2. GPU + CPU configurations with various numbers of host threads involved in computations.

GPU performance was calculated as the inverse of the sum of data chunk CPU-GPU communication, processing and GPU-CPU result transfer times. CPU performance was calculated as the inverse of data chunk processing time on the CPU(s). Average power consumption of various configurations was measured using a hardware meter within a 10 minute period for each configuration. A bash script was used to run a particular configuration. Figure 10 shows normalized performance computed as inverse of execution time divided by average power consumption through application run for 1 GPU and various numbers of streams. Normalization of performance was done by dividing results by quotients of compute time to host-device communication time ratios of various configurations. It can be seen that normalized performance per power consumption has its maxima depending on compute time to host-device communication time ratio. It is interesting to note that for 2+ numbers of streams the best normalized ratios are observed for compute time to host-device communication time ratio 9.98 and lower for the other ratios.

Furthermore, the performance by power consumption is shown for 1 and 2 GPU configurations with addition of various numbers of host threads used for computations using testbed 1. The results for the GPU/CPU performance ratio of around 30 are shown in Figure 11. It can be seen that, while execution times slightly decrease, as shown in Figure 8 before, the performance-power consumption ratio goes down due to too little improvement of execution times thanks to CPU compared to its power consumption. Had the computational power been better compared to GPUs, the ratio would have been better for higher numbers of host threads. Such a simulation was performed and its results are shown in Figure 12 for a smaller GPU/CPU relative performance ratio. It can be seen from the tests that for GPU/CPU performance equal to 2 using more host threads offers benefits in terms of performance/power consumption.

4.7 Scaling Across Multiple GPUs

The following experiments demonstrate how the code scales across GPUs in testbed 3 with 4 NVIDIA Tesla V100 Volta series GPU cards. Firstly, Figure 13 presents how the numbers of streams affect performance for the largest configuration shown in

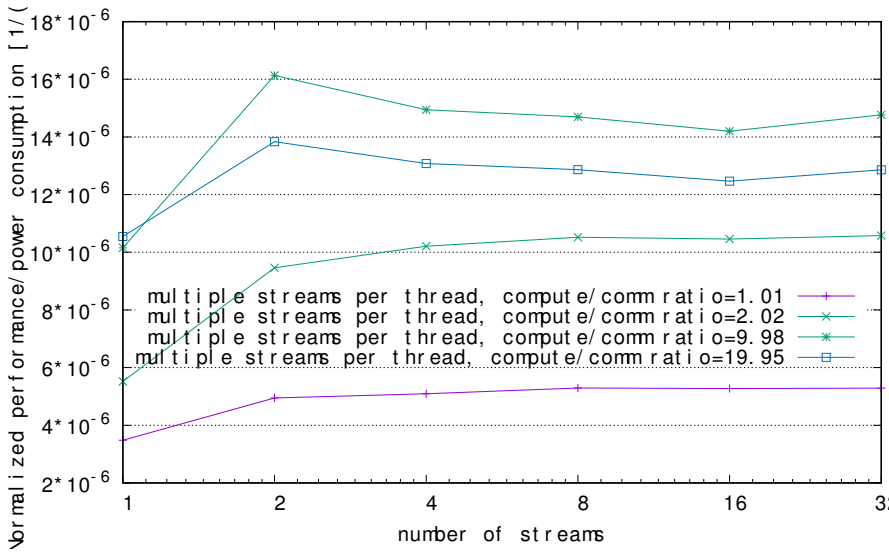


Figure 10. Normalized performance by power consumption for 1 GPU and various numbers of streams, testbed 1

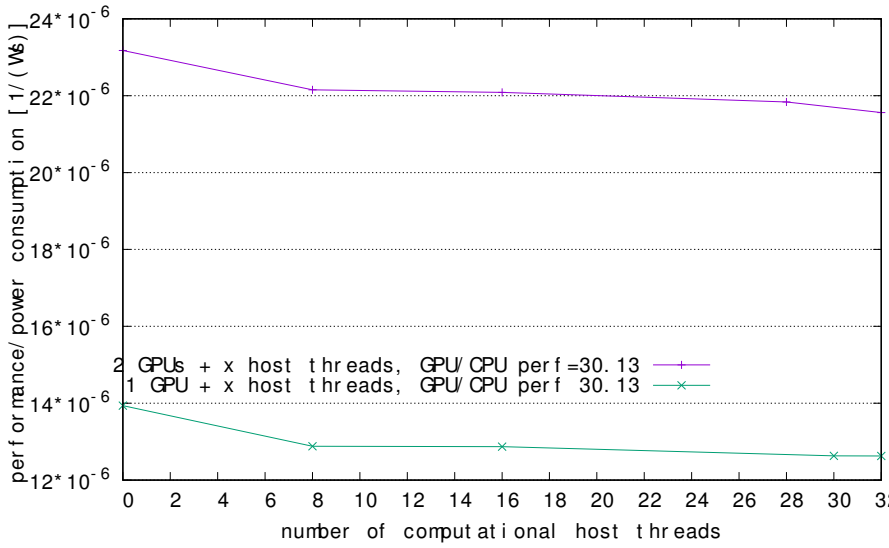


Figure 11. Performance by power consumption for 1 and 2 GPU configurations and various numbers of host threads, GPU/CPU performance = 30.13, testbed 1

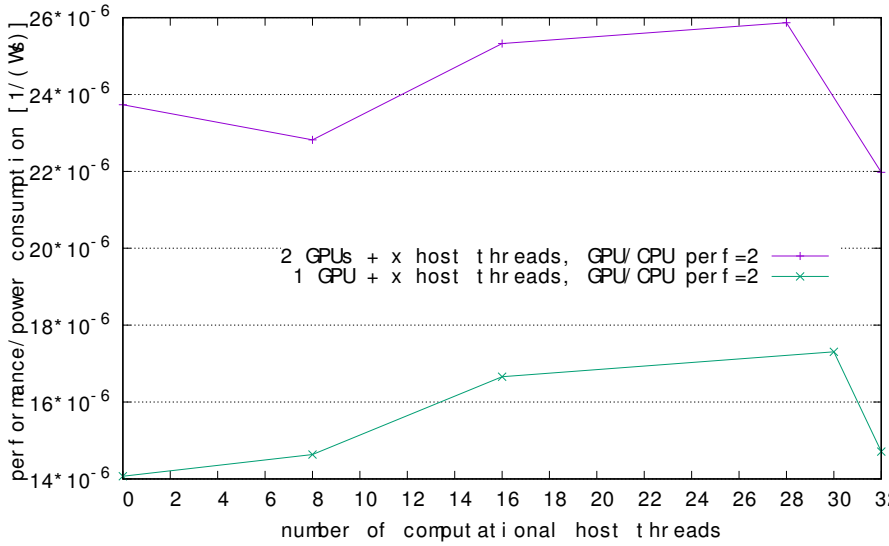


Figure 12. Performance by power consumption for 1 and 2 GPU configurations and various numbers of host threads, GPU/CPU performance = 2, testbed 1

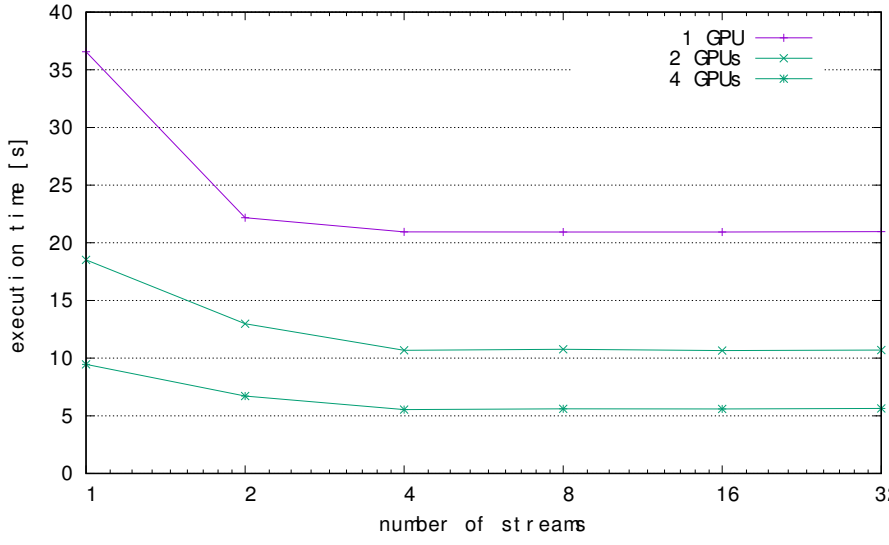


Figure 13. Execution time vs number of streams for various numbers of GPUs, largest case from Figure 2, testbed 3

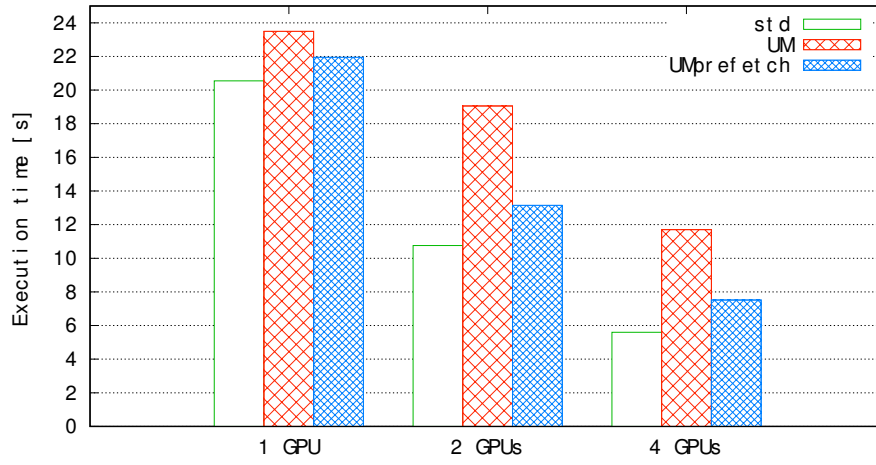


Figure 14. Execution time vs memory management solutions for various numbers of GPUs, largest case from Figure 2, 8 streams per GPU, testbed 3

Figure 2. Then, assuming 8 streams per GPU which (the configuration which already gives small execution times on the flat parts of the chart), execution times are shown for the standard memory management, UM and UMprefetch as in the previous cases. It can be seen in Figure 14 that, again, the UM version offers visible overhead over the standard memory management version. UMprefetch, thanks to manual prefetching, offers performance half-way between these two versions for 1 GPU. For 2 and 4 GPUs, it is worse than the standard memory management version by about 30 % of the difference between the other two versions.

5 CONCLUSIONS AND FUTURE WORK

In the paper we analyzed the performance and performance to power consumption ratio of multi-stream data processing on modern multicore CPU+ GPU systems. Using a benchmark that allows to set up various compute time to host-device communication time ratios, number of streams, number of threads managing computations it was possible to assess performance of various configurations on modern testbeds with Intel Xeon CPUs and NVIDIA Tesla K20m, GTX 1070 Pascal and Tesla V100 Volta series cards. The benefits of using a properly implemented multi-stream code were shown compared to GPU computations managed by various threads or processes for various numbers of streams. Furthermore, the benefits of such code compared to standard Unified Memory and Unified Memory with prefetching were shown showing performance gains at the cost of increased programming

effort. Additionally, the gains from using NVIDIA Multi Process Service have been presented for multi-process configurations. The performance to power consumption ratios have been shown for various numbers of streams and compute time to host-device communication time ratios as well as for hybrid CPU + GPU configurations for various numbers of computational threads on the host and relative GPU and CPU performances. Scalability of the code was presented between 1 and 4 GPUs using NVIDIA Tesla V100 cards.

The results can be generalized as follows. For the considered data stream processing application and various compute to communication ratios, using multiple streams, at least 2 offered visible benefits, with best results using one dedicated host thread per GPU launching communication and computations to multiple streams. Some configurations result in best execution times for 2, 4, 8 or even 16 streams but we can note that benefits over 4 streams, if any, are very small. Secondly, we confirmed that using NVIDIA MPS gives visible benefits especially for larger compute to communication ratio. Furthermore, for various compute to communication ratios we confirmed that Unified Memory brings visible overhead over the standard memory management implementation while a Unified Memory version with manual prefetching ranks between the two. For CPU + GPU codes, increasing the number of computational host threads up to the number of available logical processors decreases application execution time at rates very much depending on GPU to CPU performances with considerable gains with CPU performance in the same order as the one of the GPU. It has been shown that the observed performance per power consumption varies with the number of streams, GPU to CPU performance ratio and the number of computational host threads.

These results can be used as guidelines for best performance implementations for various applications as the tests are of generic nature and, depending on values of particular aforementioned parameters, are representative of many applications. Specifically, obtained results can be used for implementation of building blocks for data stream frameworks using multi-core CPUs and GPUs, especially multi CUDA stream communication optimization.

Future work includes extending the scope of the conducted tests performed on systems with NVIDIA Tesla V100, specifically regarding various CPU + GPU configurations, tests for various compute/communication ratios, as well as extending tests to larger V100 based systems such as NVIDIA DGX-1 featuring 8 V100 GPUs. More experiments with thread affinities will be conducted, with research of their impact for particular codes. Additionally, we plan to incorporate the outcomes of this work into higher level frameworks such as KernelHive [42] and possibly others and investigate the impact of Unified Memory oversubscription compared to the traditional implementation model.

Acknowledgments

The research in the paper has been partially supported by the Statutory Funds of Electronics, Telecommunications and Informatics Faculty, Gdansk University of

Technology, Poland. Additionally, the author would like to express his gratitude to Aleksandra Preiss from Gdansk University of Technology.

REFERENCES

[1] Aldinucci, M.—Campa, S.—Danelutto, M.—Kilpatrick, P.—Torquati, M.: Targeting Distributed Systems in Fastflow. In: Caragiannis, I. et al. (Eds.): Euro-Par 2012: Parallel Processing Workshops. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7640, 2013, pp. 47–56, doi: 10.1007/978-3-642-36949-0_7.

[2] Aldinucci, M.—Danelutto, M.—Kilpatrick, P.—Torquati, M.: Fastflow: High-Level and Efficient Streaming on Multicore. Chapter 13. In: Pllana, S., Xhafa, F. (Eds.): Programming Multi-Core and Many-Core Computing Systems. Wiley-Blackwell, 2017, pp. 261–280, doi: 10.1002/9781119332015.ch13.

[3] Bingmann, T.—Axtmann, M.—Jöbstl, E.—Lamm, S.—Nguyen, H. C.—Noe, A.—Schlag, S.—Stumpp, M.—Sturm, T.—Sanders, P.: Thrill: High-Performance Algorithmic Distributed Batch Data Processing with C++. 2016 IEEE International Conference on Big Data (Big Data), 2016, pp. 172–183, doi: 10.1109/BigData.2016.7840603.

[4] Blattner, T.—Keyrouz, W.—Chalfoun, J.—Stivalet, B.—Brady, M.—Zhou, S.: A Hybrid CPU-GPU System for Stitching Large Scale Optical Microscopy Images. 2014 43rd International Conference on Parallel Processing, 2014, pp. 1–9, doi: 10.1109/ICPP.2014.9.

[5] Carbone, P.—Katsifodimos, A.—Ewen, S.—Markl, V.—Haridi, S.—Tzoumas, K.: Apache Flink™: Stream and Batch Processing in a Single Engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Vol. 38, 2015, No. 4, pp. 28–38.

[6] Cardellini, V.—Grassi, V.—Presti, F. L.—Nardelli, M.: On QoS-Aware Scheduling of Data Stream Applications over Fog Computing Infrastructures. 2015 IEEE Symposium on Computers and Communication (ISCC), 2015, pp. 271–276, doi: 10.1109/ISCC.2015.7405527.

[7] Chen, Z.—Xu, J.—Tang, J.—Kwiat, K. A.—Kamhoua, C. A.—Wang, C.: GPU-Accelerated High-Throughput Online Stream Data Processing. IEEE Transactions on Big Data, Vol. 4, 2018, No. 2, pp. 191–202, doi: 10.1109/TBDATA.2016.2616116.

[8] Cruz, R.—Drummond, L.—Clua, E.—Bentes, C.: Analyzing and Estimating the Performance of Concurrent Kernels Execution on GPUs. XVIII Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD 2017), 2017, pp. 136–147.

[9] Cullinan, C.—Wyant, C.—Frattesi, T.: Computing Performance Benchmarks among CPU, GPU, and FPGA. Worcester Polytechnic Institute, E-project-030212-123508, 2012. https://web.wpi.edu/Pubs/E-project/Available/E-project-030212-123508/unrestricted/Benchmarking_Final.pdf.

[10] Czarnul, P.: A Model, Design, and Implementation of an Efficient Multithreaded Workflow Execution Engine with Data Streaming, Caching, and Storage Con-

straints. The Journal of Supercomputing, Vol. 63, 2013, No. 3, pp. 919–945, doi: 10.1007/s11227-012-0837-z.

[11] Czarnul, P.: Benchmarking Performance of a Hybrid Intel Xeon/Xeon Phi System for Parallel Computation of Similarity Measures Between Large Vectors. International Journal of Parallel Programming, Vol. 45, 2017, No. 5, pp. 1091–1107, doi: 10.1007/s10766-016-0455-0.

[12] Czarnul, P.: Benchmarking Overlapping Communication and Computations with Multiple Streams for Modern GPUs. In: Ganzha, M., Maciaszek, L., Paprzycki, M. (Eds.): Communication Papers of the 2018 Federated Conference on Computer Science and Information Systems (FedCSIS 2018), Poznań, Poland, 2018. Annals of Computer Science and Information Systems, Vol. 17, 2018, pp. 105–110, doi: 10.15439/2018F17.

[13] Czarnul, P.: Parallel Programming for Modern High Performance Computing Systems. 1st Edition. Chapman and Hall/CRC, Taylor & Francis, 2018. ISBN: 978-1138305953.

[14] Czarnul, P.: Parallelization of Large Vector Similarity Computations in a Hybrid CPU + GPU Environment. The Journal of Supercomputing, Vol. 74, 2018, No. 2, pp. 768–786, doi: 10.1007/s11227-017-2159-7.

[15] Czarnul, P.—Proficz, J.—Drypczewski, K.: Survey of Methodologies, Approaches, and Challenges in Parallel Programming Using High Performance Computing Systems. Scientific Programming, Vol. 2020, 2020, Art. No. 4176794, 19 pp., doi: 10.1155/2020/4176794.

[16] Danelutto, M.—De Sensi, D.—Torquati, M.: A Power-Aware, Self-Adaptive Macro Data Flow Framework. Parallel Processing Letters, Vol. 27, 2017, No. 1, Art. No. 1740004, doi: 10.1142/S0129626417400047.

[17] Dongarra, J.: Challenges for Exascale Computing. PARA 2010: State of the Art in Scientific and Parallel Computing, Reykjavík, Iceland, June 2010. `http://www.netlib.org/utk/people/JackDongarra/SLIDES/para-06102.pdf`.

[18] Friedman, E.—Tzoumas, K.: Introduction to Apache Flink: Stream Processing for Real Time and Beyond. 1st Edition. O'Reilly Media, Inc., 2016.

[19] Guo, P.—Zhang, C.: Performance Optimization for SpMV on Multi-GPU Systems Using Threads and Multiple Streams. 2016 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW), 2016, pp. 67–72, doi: 10.1109/SBAC-PADW.2016.20.

[20] Gómez-Luna, J.—González-Linares, J. M.—Benavides, J. I.—Guil, N.: Performance Models for Asynchronous Data Transfers on Consumer Graphics Processing Units. Journal of Parallel and Distributed Computing, Vol. 72, 2012, No. 9, pp. 1117–1126, doi: 10.1016/j.jpdc.2011.07.011.

[21] Jain, A.: Mastering Apache Storm: Real-Time Big Data Streaming Using Kafka, Hbase and Redis. Packt Publishing, 2017.

[22] Jarząbek, Ł.—Czarnul, P.: Performance Evaluation of Unified Memory and Dynamic Parallelism for Selected Parallel CUDA Applications. The Journal of Supercomputing, Vol. 73, 2017, No. 12, pp. 5378–5401, doi: 10.1007/s11227-017-2091-x.

[23] Knap, M.—Czarnul, P.: Performance Evaluation of Unified Memory with Prefetching and Oversubscription for Selected Parallel CUDA Applications on NVIDIA Pascal and Volta GPUs. The Journal of Supercomputing, Vol. 75, 2019, No. 11, pp. 7625–7645, doi: 10.1007/s11227-019-02966-8.

[24] Ko, B.—Han, S.—Park, Y.—Jeon, M.—Lee, B.: A Comparative Study of Programming Environments Exploiting Heterogeneous Systems. IEEE Access, Vol. 5, 2017, pp. 10081–10092, doi: 10.1109/ACCESS.2017.2708738.

[25] Kreutz, J.: CUDA Streams, Events and Asynchronous Memory Copies. April 2017. GPU Programming@Jülich Supercomputing Centre, https://www.fz-juelich.de/SharedDocs/Downloads/IAS/JSC/EN/slides/cuda/09-cuda-streams-events.pdf?__blob=publicationFile.

[26] Lee, C.-L.—Lin, Y.-S.—Chen, Y.-C.: A Hybrid CPU/GPU Pattern-Matching Algorithm for Deep Packet Inspection. PLoS ONE, Vol. 10, 2015, No. 10, Art. No. e0139301, 22 pp., doi: 10.1371/journal.pone.0139301.

[27] Lei, G.—Dou, Y.—Wan, W.—Xia, F.—Li, R.—Ma, M.—Zou, D.: CPU-GPU Hybrid Accelerating the Zuker Algorithm for RNA Secondary Structure Prediction Applications. BMC Genomics, Vol. 13, 2012, No. S-1, Art. No. S14, doi: 10.1186/1471-2164-13-S1-S14.

[28] Leibiusky, J.—Eisbruch, G.—Simonassi, D.: Getting Started with Storm. O'Reilly Media, Inc., 2012.

[29] Li, H.—Yu, D.—Kumar, A.—Tu, Y.-C.: Performance Modeling in CUDA Streams – A Means for High-Throughput Data Processing. 2014 IEEE International Conference on Big Data (Big Data), 2014, pp. 301–310, doi: 10.1109/BigData.2014.7004245.

[30] Li, W.—Jin, G.—Cui, X.—See, S.: An Evaluation of Unified Memory Technology on NVIDIA GPUs. 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2015, pp. 1092–1098, doi: 10.1109/CCGrid.2015.105.

[31] Luitjens, J.: CUDA Streams: Best Practices and Common Pitfalls. nVidia, GPU Technology Conference, 2014. http://on-demand.gputechconf.com/gtc/2014/presentations/S4158-cuda-streams-best-practices-common-pitfalls.pdf.

[32] Marks, M.—Jantura, J.—Niewiadomska-Szynkiewicz, E.—Strzelczyk, P.—Gozdz, K.: Heterogeneous GPU & CPU Cluster for High Performance Computing in Cryptography. Computer Science, Vol. 13, 2012, No. 2, pp. 63–79, doi: 10.7494/csci.2012.13.2.63.

[33] Misale, C.—Drocco, M.—Tremblay, G.—Aldinucci, M.: PiCo: A Novel Approach to Stream Data Analytics. In: Heras, D. B. et al. (Eds.): Euro-Par 2017: Parallel Processing Workshops. Springer, Cham, Lecture Notes in Computer Science, Vol. 10659, 2018, pp. 118–128, doi: 10.1007/978-3-319-75178-8_10.

[34] nVidia. CUDA Toolkit v10.0.130 Programming Guide. October 2018, https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html.

[35] Picos, K.—Díaz-Ramírez, V. H.—Tapia, J. J.: Real-Time 3D Video Processing Using Multi-Stream GPU Parallel Computing. Research in Computing Science, Vol. 80, 2014, pp. 87–95.

[36] Pinnecke, M.—Broneske, D.—Saake, G.: Toward GPU Accelerated Data Stream Processing. In: Saake, G., Broneske, D., Dorok, S., Meister, A. (Eds.): Proceedings of the 27th GI-Workshop Grundlagen von Datenbanken (GvD 2015), Gommern, Germany, May 26–29, 2015, CEUR Workshop Proceedings, CEUR-WS.org, Vol. 1366, 2015, pp. 78–83.

[37] Pirjan, A.—Petrosanu, D.-M.: Improving Parallel Programming in the Compute Unified Device Architecture Using the Unified Memory Feature. Journal of Information Systems and Operations Management, Vol. 8, 2014, No. 2, pp. 352–362.

[38] Pryor Jr., A.—Ophus, C.—Miao, J.: A Streaming Multi-GPU Implementation of Image Simulation Algorithms for Scanning Transmission Electron Microscopy. Advanced Structural and Chemical Imaging, Vol. 3, 2017, No. 1, Art. No. 15, doi: 10.1186/s40679-017-0048-z.

[39] Radford, D.—Calvert, D.: A Comparative Analysis of the Performance of Scalable Parallel Patterns Applied to Genetic Algorithms and Configured for NVIDIA GPUs. Procedia Computer Science, Vol. 114, 2017, pp. 65–72, doi: 10.1016/j.procs.2017.09.009.

[40] Rathore, M. M.—Son, H.—Ahmad, A.—Paul, A.—Jeon, G.: Real-Time Big Data Stream Processing Using GPU with Spark Over Hadoop Ecosystem. International Journal of Parallel Programming, Vol. 46, 2018, No. 3, pp. 630–646, doi: 10.1007/s10766-017-0513-2.

[41] Rennich, S.: CUDA C/C++. Streams and Concurrency, 2011, NVIDIA, `http://on-demand.gputechconf.com/gtc-express/2011/presentations/StreamsAndConcurrencyWebinar.pdf`, accessed on July 19, 2017.

[42] Rościszewski, P.—Czarnul, P.—Lewandowski, R.—Schally-Kacprzak, M.: KernelHive: A New Workflow-Based Framework for Multilevel High Performance Computing Using Clusters and Workstations with CPUs and GPUs. Concurrency and Computation: Practice and Experience, Vol. 28, 2016, No. 9, pp. 2586–2607, doi: 10.1002/cpe.3719.

[43] Sourouri, M.—Gillberg, T.—Baden, S. B.—Cai, X.: Effective Multi-GPU Communication Using Multiple CUDA Streams and Threads. 2014 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), 2014, pp. 981–986, doi: 10.1109/PADSW.2014.7097919.

[44] Verner, U.—Schuster, A.—Silberstein, M.—Mendelson, A.: Scheduling Processing of Real-Time Data Streams on Heterogeneous Multi-GPU Systems. Proceedings of the 5th Annual International Systems and Storage Conference (SYSTOR '12), 2012, Art. No. 8, 12 pp., doi: 10.1145/2367589.2367596.

[45] Wang, L.—Huang, M.—El-Ghazawi, T.: Exploiting Concurrent Kernel Execution on Graphic Processing Units. 2011 International Conference on High Performance Computing and Simulation, 2011, pp. 24–32, doi: 10.1109/HPCSim.2011.5999803.

[46] Wang, Y.—Wang, F.—Li, R.—Dou, Y.: An Efficient CPU-GPU Hybrid Parallel Implementation for DVB-RCS2 Receiver. Concurrency and Computation: Practice and Experience, Vol. 30, 2018, No. 19, Art. No. e4529, 14 pp., doi: 10.1002/cpe.4529.

[47] Zaharia, M.—Xin, R. S.—Wendell, P.—Das, T.—Armbrust, M.—Dave, A.—Meng, X.—Rosen, J.—Venkataraman, S.—Franklin, M. J.—

Ghodsi, A.—Gonzalez, J.—Shenker, S.—Stoica, I.: Apache Spark: A Unified Engine for Big Data Processing. Communications of the ACM, Vol. 59, 2016, No. 11, pp. 56–65, doi: 10.1145/2934664.

[48] Zhang, Y.—Mueller, F.: GStream: A General-Purpose Data Streaming Framework on GPU Clusters. 2011 International Conference on Parallel Processing (ICPP), 2011, pp. 245–254, doi: 10.1109/ICPP.2011.22.

Paweł Czarnul received his Ph.D. in computer science in 2003 and D.Sc. in computer science in 2015, both from the Gdansk University of Technology, Poland. His research interests include: high performance computing, distributed information systems and processing, artificial intelligence. He is author of over 80 publications in the area of parallel and distributed processing, including book entitled Parallel Programming for Modern High Performance Computing Systems, Chapman and Hall/CRC, 2018. He is currently Head of Computer Architecture Department and Vice Dean of the Faculty of ETI, Gdansk University of Technology, Poland.

Computing and Informatics, Vol. 39, 2020, 537–567, doi: 10.31577/cai_2020_3_537

IMPROVED K-ANONYMIZE AND L-DIVERSE APPROACH FOR PRIVACY PRESERVING BIG DATA PUBLISHING USING MPSEC DATASET

Priyank Jain, Manasi Gyanchandani, Nilay Khare

Maulana Azad National Institute of Technology
Bhopal MP, India
e-mail: priyankjain88@gmail.com

Abstract. Data exposure and privacy violations may happen when data is exchanged between organizations. Data anonymization gives promising results for limiting such dangers. In order to maintain privacy, different methods of k-anonymization and l-diversity have been widely used. But for larger datasets, the results are not very promising. The main problem with existing anonymization algorithms is high information loss and high running time. To overcome this problem, this paper proposes new models, namely Improved k-Anonymization (IKA) and Improved l-Diversity (ILD). IKA model takes large k-value using a symmetric as well as an asymmetric anonymizing algorithm. Then IKA is further categorized into Improved Symmetric k-Anonymization (ISKA) and Improved Asymmetric k-Anonymization (IAKA). After anonymizing data using IKA, ILD model is used to increase privacy. ILD will make the data more diverse and thereby increasing privacy. This paper presents the implementation of the proposed IKA and ILD model using real-time big candidate election dataset, which is acquired from the Madhya Pradesh State Election Commission, India (MPSEC) along with Apache Storm. This paper also compares the proposed model with existing algorithms, i.e. Fast clustering-based Anonymization for Data Streams (FADS), Fast Anonymization for Data Stream (FAST), Map Reduce Anonymization (MRA) and Scalable k-Anonymization (SKA). The experimental results show that the proposed models IKA and ILD have remarkable improvement of information loss and significantly enhanced the performance in terms of running time over the existing approaches along with maintaining the privacy-utility trade-off.

Keywords: Big data, privacy, k-anonymization, l-diversity, multi dimensional generalization, improved symmetric and asymmetric k-anonymization, Apache Storm, MPSEC dataset

1 INTRODUCTION

Data analytics and data stockpiling process is mostly connected with big data. Presently there is an exponential growth of information, which is gathered, put away, and passed on within organizations and over the web. The sudden ascent of information has brought interest towards big data use, analytics, and raised scholastic intrigue. A brief check of google trends on the search interest has been an overall increase in activity since January 2011, with maximum attention being reached around October 2017, as shown in Figure 1. Big data is considered having massive information volume and complex information structures [1]. Few illustrations of big data are social and business site information, cell phone call records, geological data, web search tool information, smart card information, and so forth.

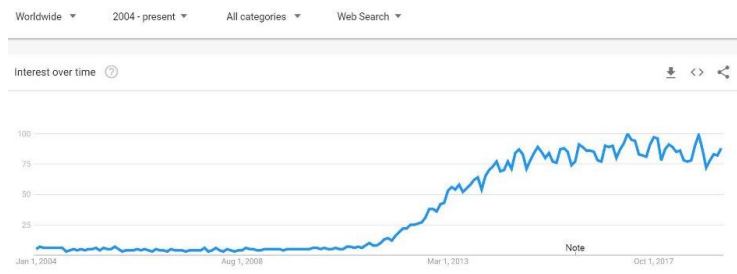


Figure 1. Search popularity of big data, source: Google trends (January 31, 2019)

Big data gives us benefits in various fields, for example in medicine, biology [2], banking [3], and social websites, and so on where a huge amount of data is collected. Now challenges are rising regarding its privacy and usage. One of the noteworthy utilization of big data use is offering data to various associations and analysts to comprehend social changes and make forecasts [4]. Approved associations, for example government organizations, banking sectors, medicinal research, have sensitive attributes in their dataset, where data distributing may cause data leakage for research purposes [5]. Differential privacy [6, 7, 21] with its expansions [8] seemed ten years back, which drives another bearing for protection saving. Subsequently, protection and privacy have to turn into an overall issue for present-day scientists. The significant issue in such a distribution is the disclosure of sensitive information, which is very bothersome [9]. Therefore, before the distribution of this information, adequate caution must be taken to conceal the sensitive details. To accomplish this, there should be a balance between privacy and utility of information. For this various algorithms has been proposed, but for a smaller dataset, like k-anonymization [16, 19, 20, 36], l-diversity [22] and t-closeness [10]. All these algorithms are based on the basic assumption that the records are free from each other and anonymization can be entirely autonomous. The widely used algorithm for data anonymization is k-anonymity [11]. For example, consider the

multidimensional patient dataset shown in Table 1 that contains personal data of the patient. The personal data of the patient consists of four disjoint sets of data: explicit identifier (EI), quasi-identifiers (QI), sensitive data (SD), and non-sensitive data (NSD). In Table 1, EI attribute is name, QI attributes are age, pin code, SD attribute is a disease and NSD attribute is job. QI are those which alone cannot provide information about an individual, but when QI is linked with the external information, it can recognize the individual by connecting them. Generalization and suppression play a crucial role in anonymization. Generalization is a technique of replacing more specific values with generic and semantically similar values. Generalization can be applied at cell or the tuple or the attribute levels. Generalization uses the concept of the domain generalization and value generalization. Each attribute in the multidimensional database is a domain. In suppression, quasi-identifiers are replaced by *, and thereby it increases the privacy of database. Thus the size of the database and content of the database is reduced. k-anonymity checks that if one record in the dataset has some value of QI then at least $k-1$ other records also have the same QI values [12, 17, 18]. The equivalency among the data tries to maintain anonymity by k times [10]. Table 2 represents patient dataset after anonymization. As an instance of patient C = $\langle$"Carl"; 52; "flu"$\rangle$, this instance is generalized and suppressed to gc = $\langle *; [50-60]$; "Respiratory infection"$\rangle$.

Name	Age	Pincode	Job	Disease
Anand	45	400052	Writer	Flu
Bharti	47	400058	Writer	Pneumonia
Carl	52	400032	Lawyer	Flu
Diana	53	400045	Artist	Stomach ulcers
Emily	64	100032	Lawyer	Stomach infection
Fatima	67	100053	Lawyer	Hepatitis
Garvin	62	200045	Writer	Stomach cancer

Table 1. Patient table

Name	Age	Pincode	Job	Disease
*	40 > && > 50	40****	Writer	Respiratory infection
*	40 > && > 50	40****	Writer	Illness
*	50 > && > 60	40****	Lawyer	Respiratory infection
*	50 > && > 60	40****	Artist	Stomach disease
*	60 > && > 70	10****	Lawyer	Stomach disease
*	60 > && > 70	10****	Lawyer	Liver disease
*	60 > && > 70	20****	Writer	Illness

Table 2. After anonymization of patient table

One of the essential clarifications for the big data find the difficulty in k-anonymization. It works on a single-dimensional function [1]. The k-anonymity and l-diversity follow one group for all information, which significantly diminishes the

obtained information, and sometimes the anonymized data is not replaced by an immediate parent; instead, it is replaced by a super parent. Implementing k-anonymity generalization in big dataset gives weak anonymization.

The multi-dimensional operation is supported by top-down generalization (TDG). The TDG strategy was proposed because of LKC privacy where L, K, C are thresholds [28]. That is used for centralization and distributed anonymization in multi-dimensional activity [13]. So applying multi-dimensional operation on it becomes multi-dimensional top-down generalization (MDTDG) [14, 15]. As a major task of anonymization, all k-anonymity methods implement the grouping process. Information typically assembled into proportionate or comparative records, known as compressions. The information loss rate decreases by using this technique.

Zakerzadeh et al. [30] introduced a new cluster-based algorithm for anonymizing numerical data streams using window processing known as fast anonymizing algorithm for numerical streaming data (FAANST). The main drawback of FAANST is that some tuples may remain in the system more than allowable time constraint. In addition, the time complexity of the algorithm is $O(n^2)$ and not efficient for data streaming [7]. Another weakness of FAANST is that it does not support categorical data. To remove this drawback another algorithm was introduced by Guo et al., FADS algorithm [31] for data stream anonymization, in which the time complexity of the approaches is $O(s)$, which is linear to the stream size s, also the space complexity is $O(c)$, which is constrained by a constant c. The main drawback of the FADS is that the algorithm does not check the remaining time of tuples that hold in the buffer in each round and that are outputted once they are probably taken into consideration to have expired. The other critical weakness of FADS is that it is not parallel and cannot handle a large number of data streams in tolerable time. Mohammadian et al. proposed FAST [32] to overcome the drawbacks of FADS. FAST protects the privacy of big data stream using parallel anonymization algorithm. It speeds up anonymization of data streams. A proactive heuristic approach was proposed in order to publish data before a specific expiration time passed. Proactive time expiration heuristic is applied to publish data before they are being expired. It works efficiently on a smaller dataset. Drawbacks of the FAST algorithm is that for the larger dataset, it results in high information loss, and the time complexity of the algorithm is comparatively high, i.e. $O(n \log n)$. Another drawback of this algorithm is that for anonymization purpose it takes super parent node for replacement instead of the current parent node to enhance running time which also causes high information loss. Zakerzadeh et al. [35] discusses the multidimensional k-anonymization Mondrian algorithm [34] and then proposes an anonymization technique for MapReduce framework: MRA. They proposed two versions of MRA. In the first version, a single global file is shared between all the nodes. The size of this file becomes larger and larger after each iteration as each node uses the same global file to update the equivalence class after each iteration. In the second version, there is no shared global file, but instead, it generates chunks of files distributed among all the nodes. Multiple iterations and file management are the major drawbacks of this technique, and as the number of iterations increases the performance decreases. In

addition, the time complexity of the algorithm is $O(n^2)$. To overcome this, Mehta et al. proposed an SKA approach using MapReduce [36]. SKA divides the input dataset into smaller equivalence classes based on all the attributes of the dataset. Classes are merged gradually (one at a time) in order to make it large, enough to fulfill k-anonymity condition. These steps were repeated for all classes. SKA takes advantage of Hadoop's data distribution (Map) phase in the class division and sort and shuffling phase in class merging; hence, it works with lesser number of iterations, compared with the existing approach [35]. The time complexity of the algorithm is $O(n \log n)$. Lack of diversity and high information loss are the major drawbacks of this work.

As per the literature review of various big data privacy mechanisms, it is observed that existing privacy mechanisms are suffering the issues of high information loss and high running time for big data. Privacy on streaming data is still a challenge and needs to be solved. Thus in this work, the focus is on the development of privacy-preserving mechanism to reduce information loss and to reduce the time taken for streaming/batch big data.

1.1 Contribution

1. To improve the time efficiency of privacy preservation algorithms in comparison with the existing approaches (FADS, FAST, MRA, and SKA).
2. Proposed Improved Symmetric k-Anonymization (ISKA) and proposed Improved Asymmetric k-Anonymization (IAKA) reduce the information loss in comparison with existing approaches.
3. Achieving higher k-value guaranteed the strongest privacy.
4. Achieving high data utility with the same level of privacy compared with existing approaches.

1.2 Organization of the Paper

The flow of paper after the introduction is, initially, Section 2 discusses the proposed model. Section 3 presents an understanding of different datasets which are used in the experiment, Section 4 covers results and discussion, and Section 5 concludes the paper with a future scope.

2 PROPOSED MODEL

Data protection is a key factor; everyone wants data to be secured as far as privacy would not incline towards losing the prominence of information [23, 24, 25, 27]. Privacy here implies hiding the actual data in such a way that analytics operations can still be performed on the data but without losing the utility of data. The privacy breach of users in any organization can be prevented using the proposed IKA and

ILD model. This model can be used for both batch and streaming dataset. The results obtained using this model are more optimized in terms of running time and information loss as compared to that of the results obtained using the existing FADS, FAST, MRA, and SKA algorithms. From protection and security point of view, it guarantees that data subjects (i.e., people) have maintainable control over their data. Figure 2 represents the proposed model, in the pre-processing phase of data, the data is cleaned, and all the missing values and irrelevant values are expelled. In further steps, the data is anonymized by using IKA, which is categorized into two parts ISKA and IAKA. Here higher values of k represent the strongest privacy and these algorithms also resolve the suppression issue of the information loss, i.e., the child node is directly replaced by a super parent instead of replacing it by an immediate parent. Then the anonymized dataset is diversified using proposed ILD model. ILD applied to the result obtained after anonymization so that it certifies that there are at least two or more unique sensitive values in each equivalence class with no attribute disclosure.

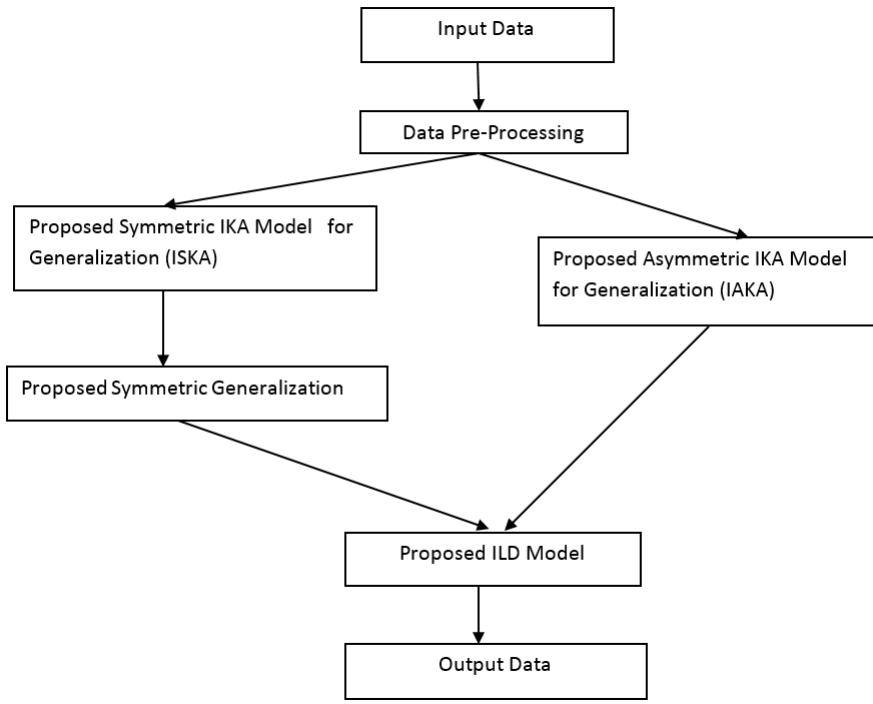


Figure 2. Proposed model

2.1 Data Pre-Processing

The data obtained contains duplicate values, additional data of the same individual or missing values. It makes the data pre-processing a vital task. The primary goal of data pre-processing is to create an appropriate analysis suitable for the dataset. Data pre-processing maintains a strategic distance from the duplicate data and the missing values as indicated by the past recorded information. Likewise, it lessens the memory and normalizes the values that are put away in a database. For achieving anonymity, there is a need to erase the identifiers and adjust the quasi-identifiers and keep the sensitive attribute. The exactness of the attributes must be considered to choose which property is a sensitive attribute, identifier, or quasi-identifiers and care must be taken to select which feature is the sensitive attribute, which is the identifier and which is a quasi-identifier. Likewise, immaterial characteristics and qualities with no significance have to be erased. In the information pre-processing, the goal is to accomplish more advancement. Few attributes should be erased as they are neither material nor essential. The first procedure in this model is to eliminate data uncertainty by using information pre-processing. By breaking down the data, there is a realization that information has no noisy value.

2.2 Proposed IKA Model

The proposed model works in the direction of falsifying the generalized data, which will make data more generalized as well as distorted. Existing work has a significant drawback of a higher degree of suppression in case of categorical attribute where values were replaced by their super parent instead of immediate parent that causes more information loss. Referring to Figure 3, the value "Local govt." should be replaced by the class of immediate parent (govt.), but instead, the superclass (work-class) is considered for generalization in the existing algorithms. There are following main reasons for this kind of high degree generalization that the individual (end-user) has specified the work-class as just local (instead of Local govt.) to the algorithm and it did not perceive it as a given workplace (because it is not matching with any of the nodes in the tree, i.e., Private, Govt. (Local gov., State gov., Federal gov.), Self emp., Without pay) so it directly went to the superclass which is "work-class" class. The proposed symmetric and asymmetric IKA model using the MDTDG technique overcomes those drawbacks. MDTDG generalizes a table which satisfies the anonymity requirement along with preserving its utility for classification. MDTDG compresses data to the topmost level, which is generalized by QI attributes [26, 29]. It takes various possible cases into account, for example if a user enters only local as its work class the algorithm will consider different possible values for same work-class (keeping account of different possible values for a single domain such as "local", "local government", "local gov" for "Local Gov." work-class domain). The information loss rate decreased by using this technique. The proposed model generalizes k-anonymity into two different types of generalization for getting accurate results.

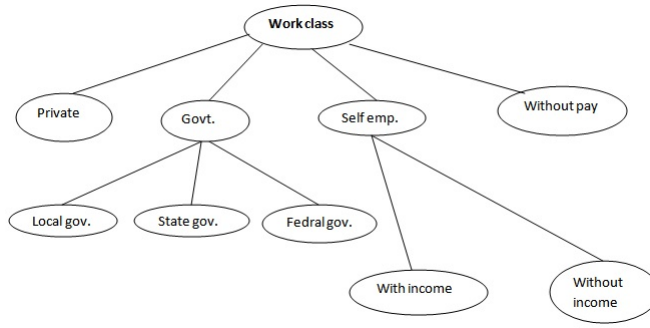


Figure 3. Taxonomy tree for work-class

2.2.1 Symmetric Anonymization

In any given dataset there is always one possible k value at a time symmetrically applied to the whole dataset. Symmetric anonymization takes equal k intervals to achieve privacy.

2.2.2 Asymmetric Anonymization

In this type of anonymization, the value of k will vary at a time. Asymmetric anonymization takes unequal k intervals to achieve privacy. The greater value of k is directly proportional to higher privacy. Asymmetric anonymization is able to achieve a higher k value as compared to symmetric anonymization. So the proposed framework endeavors to accomplish optimal k value as the higher is the k value; the more is the privacy.

The proposed Algorithm 1 is designed for both symmetric and asymmetric anonymization and tested for batch data and real-time stream data as well. The following topology used in proposed work, in which one spout (data source) and two Bolts $Bolt_1$ and $Bolt_2$ are used in *FIS* algorithm. In Figure 4 initially, input data stream s is sent into a spout that emits the data stream tuples. These tuples from spout are then sent to $Bolt_1$. It then makes Set of "delta" tuples at a time and then it inserts into Set named SOT. This set is fed as output to next bolt, i.e. $Bolt_2$. In $Bolt_2$, removal of k-anonymized clusters takes place, which is present longer than Tkc. In $Bolt_2$ function named $Publish(SOT)$ is called. After several steps, the output received from $Bolt_2$ is the k-anonymized tuples on which further processing will take place in Algorithms 2, 3 and 4.

In $Bolt_2$, pick one tuple T from SOT and publish that tuple by calling $PublishTuple()$ with SOT and T as parameters.

Algorithm 4 describes the Procedure of $PublishTuple(SOT, T)$. It is attempting to anonymize tuple T. At first, the system discovers its $k-1$ closest tuples in SOT and embeds them in the new cluster called NEW and generalizes it into $gNEW$.

Algorithm 1 Proposed IKA and ILD algorithms

INPUT: Given dataset

OUTPUT: Improved k-anonymized and l-diversity data file

1. **Step 1: cleaning(data,** A**)** // Read the data from input file row-wise in a loop
2. **If** ((len(row) < actual_row_length) **OR** ('?' in row) **OR** (' ' in row))
   Continue
   Else
   Writerow(row)
3. **Step 2a: Asymmetric_ Anonymization(**$data, A$**)**
   a1: Sort according to the values of attribute A
   a2: $FIS(S, k, \delta, Tkc, Te, NumofExecutors)$
   Goto step 5
   Or
   Step 2b: Symmetric_ Anonymization($data, A$**)**
   b1: Sort according to the values of attribute A
   b2: $K = \text{kgen}(data, A)$
   b3: $FIS(S, k, \delta, Tkc, Te, NumofExecutors)$
   Goto step 5
4. **Step 3: kgen(**$data, A$**)**
   $D =$ distinct values for attribute A
   For each value in D
   (a) Count[value] = 0
   For each row in data
   (b) Count[row[A]]+ = 1
   (c) Max = count[D[0]]
   For each value in D
   (d) Max = max(Max, count[value])
   (e) Return Max
5. **Step 5:** *ILD* **Model(data)** For each equivalence class in data {
   If every value of a sensitive attribute in an equivalence class is equal
   {
   Add tuple from next equivalence class to current equivalence class, change some values for an attribute to achieve anonymity
   }}

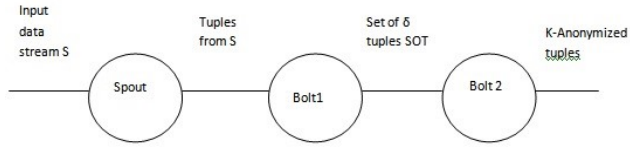


Figure 4. Topology for FIS

Algorithm 2 $FIS(S, k, \delta, Tkc, Te, NumofExecutors)$

While $|S| \neq 0$ do

1. In $Bolt_1$ δ tuples are read from *Spout* and insert them into SOT;
2. Output this SOT to $Bolt_2$;
3. In $Bolt_2$, all the clusters which exist longer than Tkc are removed;
4. Publish (SOT);

End while

Then a reusable cluster with minimum information loss *Ckbest* that covers tuple T, is chosen from SKC. If *Ckbest* exists and has smaller information loss compared to NEW, tuple T is published with *Ckbest* generalization and time of *Ckbest* is updated. Then other $k-1$ tuples that remain in SOT are checked whether they can be processed in another round or must be suppressed and published immediately.

Algorithm 3 Publish (SOT)

1. Pick the first tuple from SOT and call it T;
2. $PublishTuple(SOT, T)$;

If tuple T does not match with any cluster in SKC which has less information loss than NEW, tuple T and its neighbors are published with NEW generalization $gNEW$. Then, $gNEW$ is inserted in SKC. Alternate tuples in SOT are checked for remaining time. If they have enough time to process, they are passed to SOT otherwise they will be suppressed and published. Figure 5 represents the flow chart of a streaming algorithm.

2.3 Proposed ILD model

Another motivation behind the proposed model is to accomplish variety in the sensitive attribute. Here to achieve the privacy, information is classified. Initially, the IKA model has been applied in the dataset and then the sensitive attribute is diversified by the proposed ILD model. The proposed ILD model is an improvement of l-diversity. For each equivalence class in data value of a sensitive attribute are less

Algorithm 4 *PublishTuple*(SOT, T)

1. **Step 1:** Select k–1 unique tuple from SOT that are closest to T
   (a) Insert them into cluster NEW
   (b) Generalize NEW into $gNEW$.
2. **Step 2:** For each cluster C_k which covers T
   (a) Calculate the ILoss,
   (b) Choose a cluster with less ILoss
   (c) Call the Ck_{best} cluster.
   (d) **If** Ck_{best} **exists** and Ck_{best} produces less ILoss than $gNEW$ **then**
      i Publish T with Ck_{best} generalization;
      ii Update $round_{time}$ estimation;
      iii *Synchronized*(Ck_{best})
      { Update $Ck_{bestpublishtime}$ }
      iv **Do** in SOT for every **tuple** t
         if $(current_{time} - arrival_{time} + estimated\ round_{time}) < Te$ **then**
            Synchronized(S)
            { Insert t as the first element of S; }
         else
            Suppression and publication of t;
         end if
      end for

   else

      i Publication of NEW with $gNEW$;
      ii Update of $round_{time}$ estimation;
      iii *Synchronized*(SKC_t)
      { Insert $gNEW$ into SKC and set its time of publication; }
      iv **Do** in $(SOT - Set_{new})$ for every **tuple** t
         if $(current_{time} - arrival_{time} + estimated\ round_{time}) < Te$ **then**
            Synchronized(S)
            {
               Insert t as the first element of S;
            }
         else
            Suppression and publication of t;
         end if
      end for

   end if

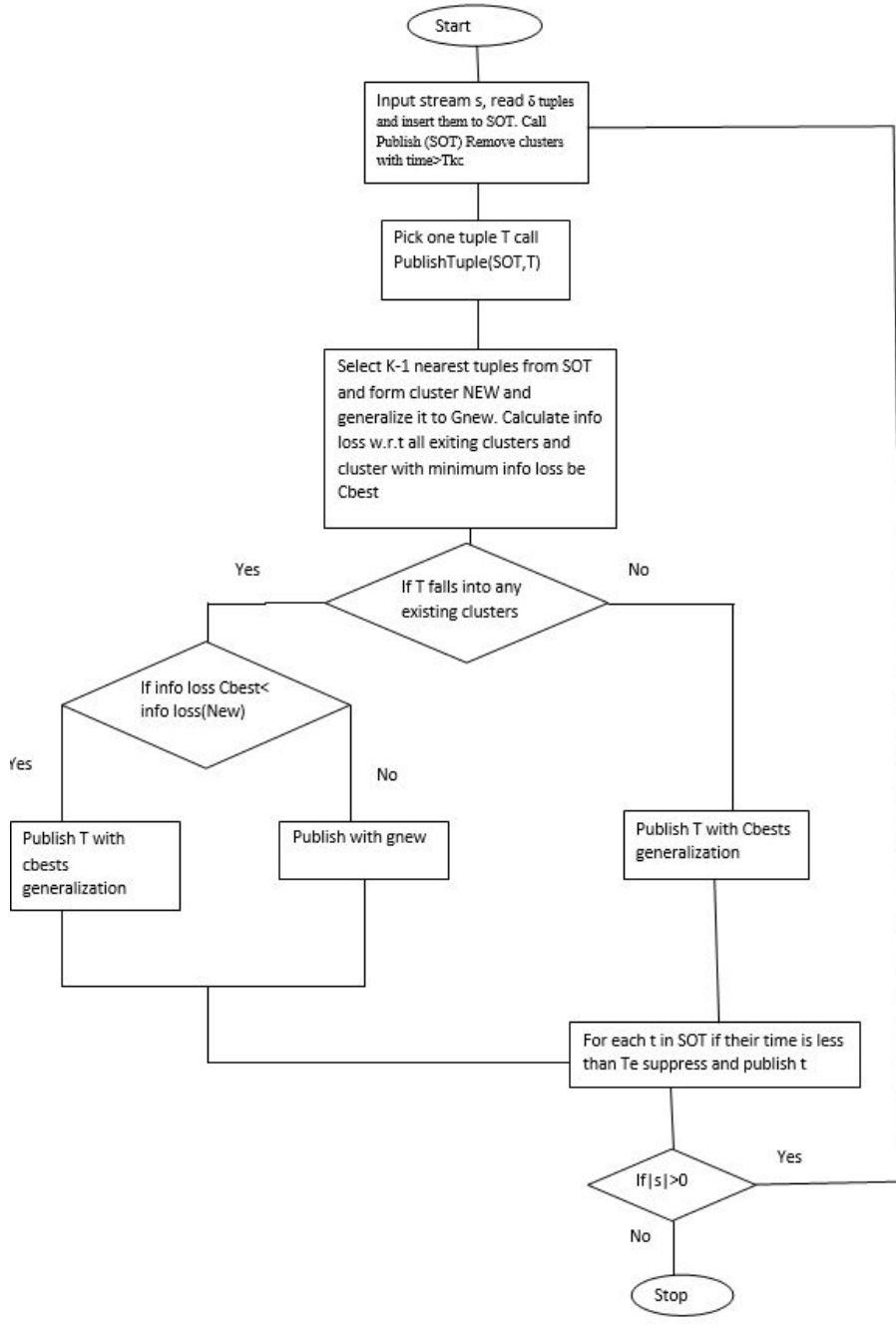


Figure 5. Flowchart for stream data (Algorithms 2, 3 and 4)

than or equal to a threshold value of l then the proposed ILD model adds a different sensitive attribute tuple from the nearest equivalence class to a current equivalence class to achieve required threshold l-diversity. The proposed ILD model decreases the probability of attribute disclosure as compared to l diversity.

Table 3 represents an example of an anonymizing dataset of healthcare, which has a sensitive attribute is the disease. In the healthcare dataset having two equivalence classes, tuple 1–5 represents one equivalence class, and tuple 6–10 represents another equivalence class in the same dataset. Equivalence class 1 represents two-diversity, and equivalence class 2 represents three-diversity in the sensitive attribute. After applying the proposed ILD model in healthcare dataset in Table 4, to maintain the required threshold of 3-diversity in each equivalence class, the proposed ILD model adds a different sensitive attribute from the nearest equivalence class to a current equivalence class. Thus, the proposed ILD model increases diversity which also increases the privacy level.

S. No.	Non-Sensitive Attributes			Sensitive Attribute
	Zip Code	Age	Nationality	Disease
1	130**	< 30	*	Cancer
2	130**	< 30	*	Cancer
3	130**	< 30	*	Corona
4	130**	< 30	*	Cancer
5	130**	< 30	*	Cancer
6	130**	3*	*	Heart Disease
7	130**	3*	*	Heart Disease
8	130**	3*	*	Cancer
9	130**	3*	*	Cancer
10	130**	3*	*	Corona

Table 3. l-diversity before ILD model

S. No.	Non-Sensitive Attributes			Sensitive Attribute
	Zip Code	Age	Nationality	Disease
1	130**	< 30	*	Cancer
2	130**	< 30	*	Cancer
3	130**	< 30	*	Corona
4	130**	< 30	*	Heart Disease
5	130**	< 30	*	Cancer
6	130**	3*	*	Heart Disease
7	130**	3*	*	Heart Disease
8	130**	3*	*	Cancer
9	130**	3*	*	Cancer
10	130**	3*	*	Corona

Table 4. l-diversity after ILD model

3 DATASET USED

For the experimental purpose, this work used three datasets: the poker dataset, adult dataset and MPSEC dataset.

3.1 Poker Dataset

The poker dataset [37], collected from UCI, has 11 numerical attributes and millions of instances. Here the first ten predictive attributes are used as quasi-identifiers and t class variable is used as the sensitive attribute.

3.2 Adult Dataset

The adult dataset [38], collected from UCI, has 14 numerical and categorical attributes and 48 842 instances. This dataset is widely used for the privacy-preserving purpose. Here the sensitive attribute in the dataset is age (numerical) and profession (categorical).

3.3 MPSEC Dataset

This is for the first time when the proposed methodology has been implemented on a newly collected dataset from Madhya Pradesh State Election Commission (MPSEC), Bhopal, India. It is state voter and candidate dataset. It consists of 34 attributes. After pre-processing, 12 useful attributes were extracted, specifically age, district code, candidate name, gender, category, mobile no., candidate designation, ward no., votes, marital status, auto-id, and occupation. In the proposed work, "Occupation" is considered a sensitive attribute. The dataset has a candidate name and mobile number as EI attributes and the rest of the attributes are QI. We find the interesting patterns from MPSEC datasets, if combined with the demographic data, the percentage of people who were eligible and voted in the elections can be calculated. The correlation and dependency between the caste of the voters and the winning candidate can also be found. The co-dependency between the female candidates and their occupations might be detected. The percentage of female candidates amongst total candidates can be calculated. If combined with Aadhar card data, the voting pattern between the reserved category and unreserved category voters can be found. The percentage of different age groups standing for election can be detected, whether it has more of the young candidates or older candidates. Table 5 represents the number of tuples and the corresponding data size of MPSEC dataset for experiment purpose.

4 RESULTS AND DISCUSSION

The platform used for the deployment is HP Z840 workstation. It consists of 64-bit dual-core processors and 8 GB of RAM. Apache storm combination with Python is

MPSEC Dataset	
Number of Tuples	**Size**
35 000	4 MB
100 000	10.5 MB
1 million	104.2 MB
10 million	1.01 GB
100 million	10.4 GB

Table 5. MPSEC dataset

used to implement the proposed algorithms in the multi-node environment. The multi-node environment created by using 5 workstations. Each workstation has 40 cores. In our experiments, 40 cores are used for the name node, and 160 cores are used for worker nodes for implementing the proposed algorithms. The parameters used for comparison are completeness, running time, and information loss. Existing methods – FADS, FAST, MRA, and SKA – are implemented in the same environment. The proposed algorithms have been applied to MPSEC dataset, adult dataset [37], and poker dataset [38]. The proposed algorithms can also be applied to any other datasets which require the privacy mechanism.

4.1 Completeness

Completeness describes whether the data is fully anonymized or not. In the asymmetric algorithm, all data is generalized in an asymmetric way so that IAKA achieves 100 percentage completeness. The value of k achieved is 1 523. In ISKA, symmetric grouping value from k is changed. If (new $k <$ gen. k) < 100 percentage completeness, data is only generalized, not anonymized, and it can be easily predictable, if (new $k >=$ gen. k). Here also, ISKA tries to achieve 100 percentage completeness, the value of k achieved is 1 283. The proposed model gives a better result with large dataset having higher k value. The higher k value guaranteed the strongest privacy.

4.2 Running Time

The running time complexity of IAKA and ISKA is described as follows, both the algorithms having mainly three functions. The first function is distance function, which is used for calculating the distance between two tuples, for finding the best nearest tuples for anonymization purpose. It is used by symmetric and asymmetric intervals for ISKA and IAKA algorithms, respectively. Distance function loop variable is incremented by a constant amount of time for both algorithms, which represents the time complexity of the distance variable being $O(n)$. The second function is the information loss function, and it is used to find the best optimal cluster, which is having minimum information loss. Similar to distance function, the information loss function is incremented by a constant amount of

time. The time complexity of the information loss function is $O(n)$. The third function checks the expiration time of tuple, whether the time since tuple arrived is less than that of proactive heuristic and generalizes them within those time limits. This function takes a constant amount of time. So the overall time complexity of the proposed algorithms IAKA and ISKA is $O(n)$. After IAKA and ISKA algorithms, the proposed ILD algorithm maintains the required threshold of diversity in each equivalence class. In the proposed ILD model, diversity function loop variable is incremented by a constant amount of time, which represents the time complexity of the diversity function being $O(n)$. So the overall time complexity of the proposed ILD algorithm is $O(n)$. The time complexity of the proposed IKA and ILD algorithms found to be $O(n)$ where n = number of tuples of a given attribute. The proposed algorithms work on a lesser number of iterations: only three iterations in the case of both the algorithms of IKA and four iterations in the case of ILD algorithm. The comparison of time complexity of the proposed algorithms with the existing methods is shown in Table 6. The proposed work and the existing methods FADS, FAST, MRA and SKA have been implemented in the same experimental environment. In the experiment the anonymity degree k varied from 10 to 640 and the diversity level l is set to 6 in this proposed work.

The IAKA and ISKA are more efficient than the existing algorithms (FADS, FAST, MRA, SKA). The disadvantage of the FADS is that the algorithm does not take a look at the remaining time of tuples that are kept within the buffer in each round and are outputted once they are probably taken into consideration to have expired. The critical weakness of FADS and FAST is that they are not able to handle a larger dataset of 10M size. The major drawbacks of MRA are multiple iterations and file management, and as the number of iterations increases the performance decreases. In addition, the time complexity of the MRA is very high, i.e. $O(n^2)$. The time complexity of the SKA algorithm is also high, i.e. $O(n \log n)$. Lack of diversity is the major drawback of SKA, which causes attribute discloser. To overcome the FADS weakness the proposed algorithms check the expiration time of tuple, whether the time since tuple arrived is less than that of proactive heuristic and generalize them within that time limits. The running time has improved due to the proposed IKA and ILD algorithms which take only fewer iterations and execute on the multi-node environment of big data. The proposed improved algorithms are efficient to handle large database and proposed ILD model maintains at least 6 diversity in each equivalence class what overcomes attribute discloser. Comparing both algorithms of IKA, IAKA is more efficient as it is taking less running time as compared to ISKA. The running time declines with the increasing number of tuples or records, mostly because fewer iterations are required to satisfy privacy requirements. Tables 7, 8, 9 show the running time of MRA, SKA, IAKA and ISKA on 1M, 10M and 100M dataset with respect to different k values, respectively. In Table 7, when the value of k is 10 on MPSEC 1M dataset then the running time of IAKA and ISKA is 335 and 328 seconds and when the value of k is 640 the running time of IAKA and ISKA is 305 and 302.3 seconds,

S. No.	Algorithms	Time Complexity	Remark
1	FADS	$O(s)$, which is linear to the stream size s also the space complexity is $O(c)$, which is constrained by a constant c.	FADS algorithm does not check the remaining time of tuples and FADS is not parallel and cannot handle a larger dataset of 10M size.
2	FAST	$O(n \log n)$	Large dataset results in high information loss and cannot handle a larger dataset of 10M size.
3	MRA (Map Reduce-based Anonymization)	$O(n^2)$	Multiple iterations and file management are the major drawbacks of this technique and as the number of iterations increases the performance degrades.
4	SKA (Scalable k-Anonymization)	$O(n \log n)$	Lack of diversity and high information loss are the major drawbacks of this work.
5	**Proposed IKA and ILD algorithms**	$O(n)$	In the proposed algorithms complexity decreases due to lesser number of iterations.

Table 6. Comparison of time complexity of proposed algorithms with competing or existing methods

respectively. In Table 8, when the value of k is 10 on MPSEC 10M dataset then the running time of IAKA and ISKA is 2 070.2 and 1 989.3 seconds and when the value of k is 640 the running time of IAKA and ISKA is 1 641.5 and 1 555.8 seconds, respectively. ISKA performs best in running time and both IAKA and ISKA algorithms are outperformed as compared to the existing MRA and SKA algorithms. When the value of k is higher, i.e. higher privacy, then running time goes down in all the algorithms due to low computational cost required, similarly to Tables 7, 8, and 9, it is representing the running time on MPSEC 100M dataset, respectively. This work also finds an interesting pattern that the running time values of on MPSEC 10M and 100M dataset are not increasing proportionally as compared to Table 7, these running time values are much smaller by using our proposed algorithms. So our proposed algorithms take less running time as compared to the existing algorithms for larger datasets. Table 10 shows the running time of FADS, FAST, IAKA and ISKA on 1M dataset in which IAKA repeatedly performs best.

Value of k	Running Time in Seconds			
	MRA	SKA	IAKA	ISKA
10	1 200	675	335	328
20	1 189	541	330.2	327.5
40	1 089	500	326.3	324.3
80	1 000	472	321.2	320
160	920	421	317.3	316.2
320	880	400	310	308.2
640	812	398	305	302.3

Table 7. Running time comparison of MRA, SKA, IAKA, and ISKA on MPSEC 1M dataset in seconds

Value of k	Running Time in Seconds			
	MRA	SKA	IAKA	ISKA
10	8 000	4 010	2 070.2	1 989.3
20	7 800	3 900	2 000.2	1 965.1
40	7 100	3 508	1 905.3	1 845.8
80	6 600	3 280	1 857.6	1 780.3
160	6 200	3 121	1 779.8	1 697.4
320	5 807	2 872	1 701	1 623.2
640	5 410	2 710	1 641.5	1 555.8

Table 8. Running time comparison of MRA, SKA, IAKA, and ISKA on MPSEC 10M dataset in seconds

4.2.1 Comparison of IAKA and ISKA Algorithms with Existing Batch Data Anonymization Algorithms MRA and SKA

The average running times on 1M, 10M, and 100M datasets are depicted in Figures 6, 7, and 8, respectively. As can be seen in the figures, both IAKA and ISKA have smaller running time than that of MRA and SKA. And SKA has smaller running time than that of MRA [35, 36] because SKA performs the task in less number

Value of k	Running Time in Seconds $*$ 10			
	MRA	SKA	IAKA	ISKA
10	7 800	3 840	1 980.2	1 909
20	7 100	3 509	1 920.5	1 875.8
40	5 200	2 690	1 797.2	1 745
80	4 100	2 150	1 791.8	1 680.6
160	4 400	2 198	1 699.2	1 588.4
320	4 120	2 098	1 600.8	1 505.6
640	3 900	1 850	1 541.8	1 432.2

Table 9. Running time comparison of MRA, SKA, IAKA, and ISKA on MPSEC 100M dataset in seconds $*$ 10

Value of k	Running Time in Seconds			
	FADS	FAST	IAKA	ISKA
10	610	422.3	335	328
20	608.5	419.5	330.2	327.5
40	609	417	326.3	324.3
80	607	415	321.2	320
160	608	411.3	317.3	316.2
320	607.5	410	310	308.2
640	607	409.3	305	302.3

Table 10. Running time comparison of FADS, FAST, IAKA, and ISKA on MPSEC 1M dataset in seconds

of iterations than that of MRA. In the case of ISKA, there will be no overhead for checking the distance between the tuples as it is concerned more with equal-sized cluster, but on the nearness degree of tuples in the dataset. On the other hand, IAKA is related more with nearness of data than with the equality of cluster sizes. This requires to calculate the distance between the tuple of interest and the generalized cluster to decide whether the tuple can be inserted in that cluster or not which results in more running time of IAKA and relatively less running time of ISKA.

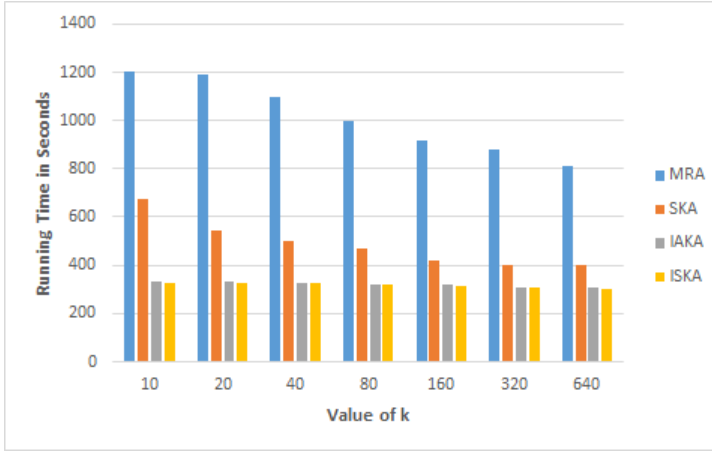


Figure 6. Running time of MRA, SKA, IAKA, and ISKA on MPSEC 1M dataset

4.2.2 Comparison of IAKA and ISKA Algorithms with Existing Stream Data Anonymization Algorithms FADS and FAST

The average running time on 1M synthetically generated MPSEC stream dataset are depicted in Figure 9. As can be seen in this figure, both IAKA and ISKA

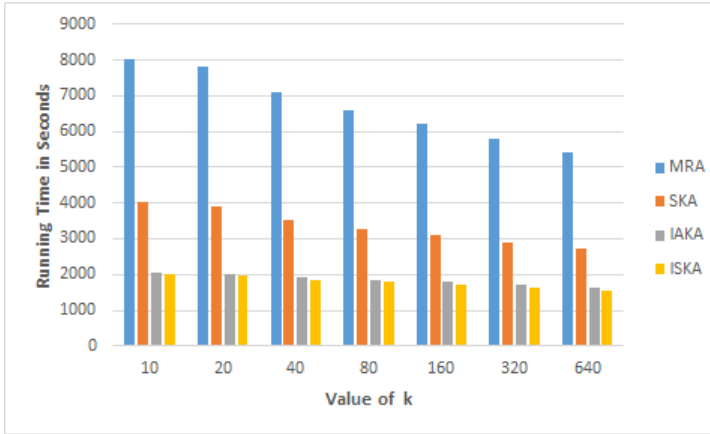


Figure 7. Running time of MRA, SKA, IAKA, and ISKA on MPSEC 10M dataset

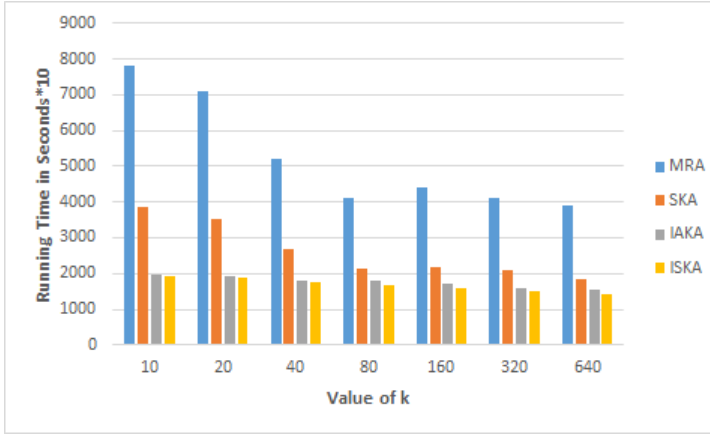


Figure 8. Running time of MRA, SKA, IAKA, and ISKA on MPSEC 100M dataset

have smaller running time than that of FADS and FAST. And FAST has smaller running time than that of FADS [31, 32]. FAST have smaller running time than that of FADS as its implementation uses the concept of multithreading. IAKA has larger running time than ISKA and the reasons are the same as mentioned in the Section 4.2.1.

4.3 Information Loss

Information loss is a term that is shown in Equation (1). In this equation, $lower_{ij}$ and $upper_{ij}$ represent lower and upper bound of attribute j in tuple i after general-

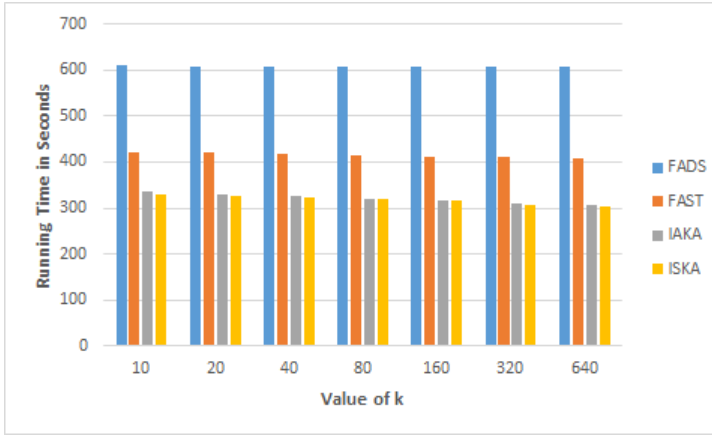


Figure 9. Running time of FADS, FAST, IAKA, and ISKA on MPSEC 1M dataset

ization, respectively, $\min_j$ and $\max_j$ represent the minimum and maximum values, respectively, taken by attribute j over all records.

$$I = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{|upper_{ij} - lower_{ij}|}{n.m|Max_j - Min_j|}. \tag{1}$$

In the experiment, the anonymity degree k varied from 10 to 640, and the diversity level l is set to 6 in this proposed work. The reported values (except for the information loss) are averaged over two runs. Tables 11, 12, 13 show the information loss of MRA, SKA, IAKA and ISKA on 1M, 10M and 100M dataset with respect to different k values, respectively. In Table 11, when the value of k is 10 on MPSEC 1M dataset then the information loss of IAKA and ISKA is 18 and 21 percentage, and when the value of k is 640 the information loss of IAKA and ISKA is 27.39 and 28.97 percent, respectively. In the case of existing algorithms MRA and SKA in the same table, when the value of k is 10 on MPSEC 1M dataset then the information loss is 33.4 and 29.84 percent, and when the value of k is 640 the information loss is 54.40 and 44.87 percent, respectively. The proposed algorithms show better performance regarding the information loss as compared to the existing methods. In Table 12, when the value of k is 10 on MPSEC 10M dataset then the information loss of IAKA and ISKA is 15.28 and 17.89 percent, and when the value of k is 640 the information loss of IAKA and ISKA is 24.18 and 25.07 percent, respectively. In the case of existing algorithms MRA and SKA in the same table, when the value of k is 10 on MPSEC 10M dataset then the information loss is 31.2 and 25.9 percent, and when the value of k is 640 the information loss is 48.80 and 41.08 percent, respectively. As compared to Table 11, data size is increasing 1M to 10M, information loss is decreasing and proposed algorithms IAKA and ISKA show incredible performance, as compared to the existing algorithms.

In Table 13, when the value of k is 10 on MPSEC 100M dataset then the information loss of IAKA and ISKA is 12.8 and 14 percent, and when the value of k is 640 the information loss of IAKA and ISKA is 21.91 and 24.58 percent, respectively. In the case of existing algorithms MRA and SKA in the same table, when the value of k is 10 on MPSEC 100M dataset then the information loss is 27.12 and 19.41 percent, and when the value of k is 640 the information loss is 45.01 and 41.47 percent, respectively. It is clear that as data size increases, information loss decreases due to the large crowd effect. It is also observed that IAKA outperforms ISKA, MRA and SKA in terms of information loss. ISKA also presents remarkable improvements in terms of information loss as compared to existing methods.

In Table 14, there is a considerable difference in the information loss of our proposed IAKA and ISKA algorithms with streaming data FADS and FAST algorithm. This is because the FADS and FAST algorithms fail to take full advantage of the entire data, because of their inability to merge the values across big data chunks. Typically, a larger difference is expected if the data is split into more chunks. Another drawback of a FAST algorithm for anonymizing data is that it takes super parent node for replacement instead of the current parent node for categorical attribute to enhance time which results in high information loss. The drawback of MRA and SKA also is high information loss. Among IAKA and ISKA, IAKA has less information loss than that of ISKA because, in ISKA, the size of each cluster has to be the same and in order to satisfy this property it sometimes has to compromise over the nearness of the tuples in the cluster which is otherwise done for less information loss in the anonymized data. On the other hand, the IAKA does not impose any condition on the size of the cluster but concentrates more on the nearness of the data in the cluster that results in less information loss. Another efficiency of proposed IAKA and ISKA of IKA model is using the MDTDG technique for categorical attributes, so information loss rate decreased. The proposed algorithms also utilize the large crowd effects, i.e. the same amount of privacy applied to larger dataset. It is also able to achieve low information loss and privacy-utility trade-off.

Value of k	Information Loss in Percentage			
	MRA	SKA	IAKA	ISKA
10	33.4	29.84	18	21
20	38.57	32.08	20	22.4
40	41.72	36.12	21.32	23.6
80	44.08	38.27	23.87	24.74
160	47.40	39.9	25.18	25.75
320	51.81	41.87	26.43	27.88
640	54.40	44.87	27.39	28.97

Table 11. Information loss comparison of MRA, SKA, IAKA, and ISKA on MPSEC 1M dataset

Value of k	Information Loss in Percentage			
	MRA	SKA	IAKA	ISKA
10	31.2	25.9	15.28	17.89
20	36.01	29.8	17.87	19.72
40	39.92	34	19.08	21
80	42.02	36	20.87	22.57
160	44.08	37.01	21.84	23.27
320	46.09	39.84	22.70	24.89
640	48.80	41.08	24.18	25.07

Table 12. Information loss comparison of MRA, SKA, IAKA, and ISKA on MPSEC 10M dataset

Value of k	Information Loss in Percentage			
	MRA	SKA	IAKA	ISKA
10	27.12	19.41	12.8	14
20	32.5	24.05	13.49	16.72
40	37.61	30.58	16.19	19.29
80	40.8	32.41	18.09	21.08
160	43.21	38.48	19.18	22.39
320	44.75	40	20.04	23.68
640	45.01	41.57	21.91	24.58

Table 13. Information loss comparison of MRA, SKA, IAKA, and ISKA on MPSEC 100M dataset

4.3.1 Comparison of IAKA and ISKA Algorithms with Existing Batch Data Anonymization Algorithms MRA and SKA

The average information loss on 1M, 10M and 100M MPSEC datasets are depicted in Figures 10, 11 and 12, respectively. As can be seen in the figures, both IAKA and ISKA outperformed SKA and MRA. SKA outperformed MRA [35, 36]. The reason is that the former algorithm uses the proactive heuristic variable to maintain

Value of k	Information Loss in Percentage			
	FADS	FAST	IAKA	ISKA
10	38	31	18	21
20	42	33.4	20	22.4
40	47	35.8	21.32	23.6
80	49	39.2	23.87	24.74
160	51	43	25.18	25.75
320	57	45.5	26.43	27.88
640	65.3	51	27.39	28.97

Table 14. Information loss comparison of FADS, FAST, IAKA, and ISKA on MPSEC 1M dataset

the relativity of the data to the situation. As the size of dataset increases the information loss decreases due to large crowd effect.

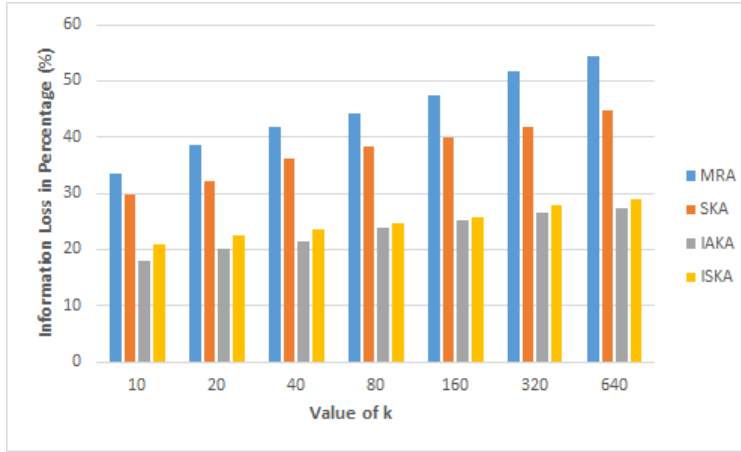


Figure 10. Information loss of MRA, SKA, IAKA, and ISKA on MPSEC 1M dataset

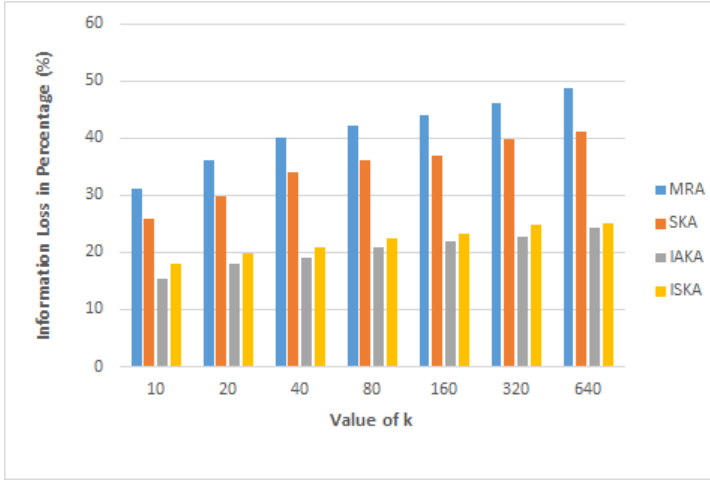


Figure 11. Information loss of MRA, SKA, IAKA, and ISKA on MPSEC 10M dataset

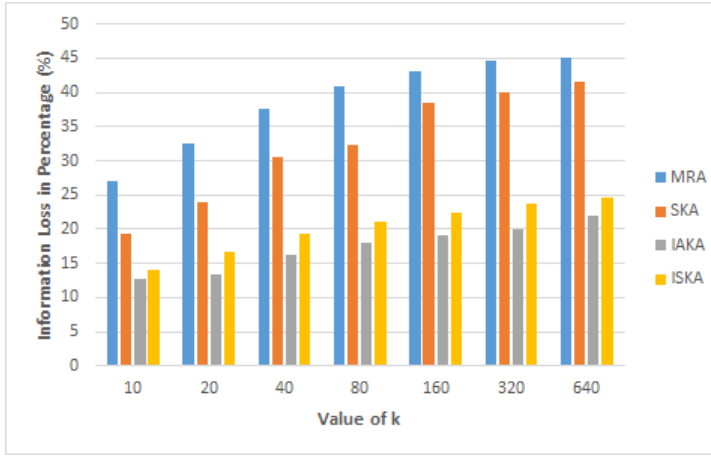


Figure 12. Information loss of MRA, SKA, IAKA, and ISKA on MPSEC 100M dataset

4.3.2 Comparison of IAKA and ISKA Algorithms with Existing Stream Data Anonymization Algorithms FADS and FAST on Synthetically Generated Stream Data

The average information loss on 1M synthetic MPSEC stream data is depicted in Figure 13. As can be seen in the figure, IAKA and ISKA perform better than FADS and FAST. FAST outperformed FADS [31, 32]. Here for stream data, we are comparing only till 1M dataset as no other previous papers have considered dataset over that size. Here also, IAKA performs better than ISKA and the reasons are the same as mentioned in Section 4.3.1.

4.4 Comparison of Proposed Algorithms Using Different Datasets

Proposed IKA and ILD algorithms have also been applied to adult dataset [37] and poker dataset [38]. Tables 15 and 16 show the comparison of proposed IAKA and ISKA algorithms used with different datasets (adult dataset, poker dataset) of 10M size. The proposed IAKA and ISKA and the existing methods, MRA and SKA, are implemented in the same experiment environment using different dataset, i.e. adult dataset and poker dataset. In this experiment, anonymity degree k is set to 80, i.e. most widely used value, and the diversity level l is set to 6. Table 15 represents the running time comparison of MRA, SKA, IAKA, and ISKA on different datasets in which the performance of ISKA is best with all three datasets. Both the proposed algorithms of IKA have an optimum time complexity, i.e. only $O(n)$, and the reason is already discussed in Section 4.2. The major drawbacks of MRA are multiple iterations and file management, and as the number of iterations increases performance decreases. In addition, the time com-

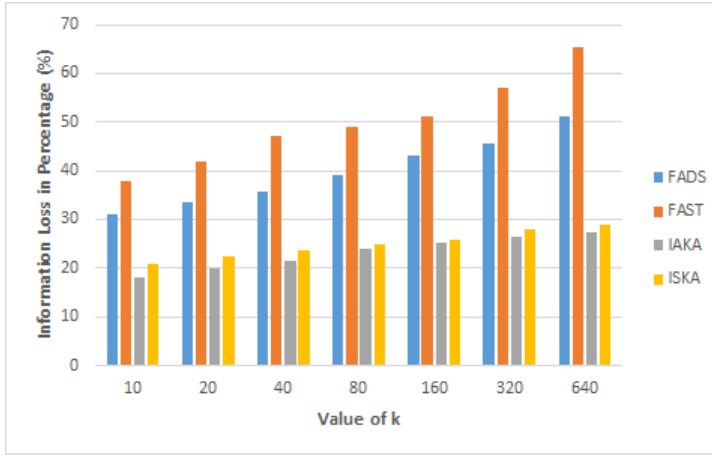


Figure 13. Information loss of FADS, FAST, IAKA, and ISKA on MPSEC 1M dataset

plexity of the MRA is very high, i.e. $O(n^2)$. The time complexity of the SKA algorithm is $O(n \log n)$. Lack of diversity and high information loss are the major drawbacks of SKA. Table 16 represents information loss comparison of MRA, SKA, IAKA, and ISKA on different datasets, in which performance of IAKA is the best with all three datasets because IAKA concentrates more on the nearness of the data in the cluster and it does not impose any condition on the size of the cluster that results in less information loss. ISKA also shows the significant reduction in information loss because both algorithms of IKA model use MDTDG technique for categorical attributes and also utilize the large crowd effects. So both algorithms of IKA are able to achieve low information loss and privacy-utility trade-off.

Name of Dataset	Running Time in Seconds			
	MRA	SKA	IAKA	ISKA
Adult Dataset [38]	7 135	3 415	1 975.2	1 916.3
Poker Dataset [37]	6 830	3 329	1 956.2	1 899.1
MPSEC Dataset	6 600	3 280	1 857.6	1 780.3

Table 15. Running time comparison of MRA, SKA, IAKA, and ISKA on different 10M size datasets in seconds where value of $k = 80$

5 CONCLUSION

This paper addresses the issue of high information loss and high running time of anonymization algorithms of big data. This paper proposed the IKA and ILD model and applied them to the MPSEC dataset and successfully achieved high k-value

Name of Dataset	Information Loss in Percentage			
	MRA	SKA	IAKA	ISKA
Adult Dataset [38]	47.31	40.16	21.95	23.89
Poker Dataset [37]	45.02	38.9	21.89	23.31
MPSEC Dataset	42.02	36	20.87	22.57

Table 16. Information loss comparison of MRA, SKA, IAKA, and ISKA on different 10M size datasets in percentage where value of $k = 80$

(k value 1 523 in asymmetric, k value 1 283 in symmetric) with the maintained diversity in the sensitive attribute. As shown in the experimental result, 100 percentage completeness has been achieved, that is the data was fully anonymized, and the time complexity is $O(n)$. ISKA proves to be more efficient since the running time is less when compared to IAKA and other existing algorithms FADS, FAST, MRA and SKA. The running time has improved because the proposed IKA and ILD algorithms takes fewer iterations and execute on the multi-node environment of big data. ISKA and IAKA algorithms have reduced remarkable information loss in comparison with the existing methods FADS, FAST in case of stream data. They are better than MRA and SKA in case of batch data and IAKA performs best regarding the information loss. The proposed IKA and ILD models maintain the privacy-utility trade-off. The improvement in this model is that rather than anonymizing batch data and streaming data differently with a bunch of algorithms, it is better to achieve the combined functionalities in the single algorithm. The proposed IKA and ILD algorithms can also be applied to any datasets which require privacy mechanism. These proposed algorithms are useful for healthcare, sensor networks, online flight reservation systems, marketing and other commercial companies to grow their business. As their database contains personal information, it is vulnerable to provide direct access to researchers and analysts. Since in this case the privacy of individuals is leaked, it can pose a threat and it is also illegal. The future work for this approach is directed towards creating a new model which deals with privacy issues of correlative data for big data publication purposes.

Acknowledgement

We are grateful to the Madhya Pradesh State Election Commission, India, for their enthusiastic and constant support and for providing us with the real-time big dataset needed for the research work. We also acknowledge the concern of the Madhya Pradesh Council of Science and Technology, Bhopal, India and providing us funds to carry out this research work.

REFERENCES

[1] Al-Zobbi, M.—Shahrestani, S.—Ruan, C.: Sensitivity-Based Anonymization of Big Data. 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops), Dubai, 2016, pp. 58–64, doi: 10.1109/LCN.2016.029.

[2] Marx, V.: The Big Challenges of Big Data. Nature, Vol. 498, 2013, pp. 255–256, doi: 10.1038/498255a.

[3] Sánchez, D.—Martínez, S.—Domingo-Ferrer, J.: Comment on "Unique in the Shopping Mall: On the Reidentifiability of Credit Card Metadata". Science, Vol. 351, 2016, No. 6279, pp. 1274–1276, doi: 10.1126/science.aad9295.

[4] Yu, S.—Guo, S. (Eds.): Big Data Concepts, Theories, and Applications. Springer, Cham, 2016, doi: 10.1007/978-3-319-27763-9.

[5] Xu, L.—Jiang, C.—Wang, J.—Yuan, J.—Ren, Y.: Information Security in Big Data: Privacy and Data Mining. IEEE Access, Vol. 2, 2014, pp. 1149–1176, doi: 10.1109/ACCESS.2014.2362522.

[6] Dwork, C.: Differential Privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (Eds.): Automata, Languages, and Programming (ICALP 2006). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4052, 2006, pp. 1–12, doi: 10.1007/11787006_1.

[7] Dwork, C.—McSherry, F.—Nissim, K.—Smith, A. D.: Calibrating Noise to Sensitivity in Private Data Analysis. In: Halevi, S., Rabin, T. (Eds.): Theory of Cryptography (TCC 2006). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3876, 2006, 2006, pp. 265–284, doi: 10.1007/11681878_14.

[8] Geng, Q.—Viswanath, P.: Optimal Noise Adding Mechanisms for Approximate Differential Privacy. IEEE Transactions Information Theory, Vol. 62, 2016, No. 2, pp. 952–969, doi: 10.1109/TIT.2015.2504972.

[9] Qu, Y.—Yu, S.—Gao, L.—Niu, J.: Big Dataset Privacy Preserving Through Sensitive Attribute-Based Grouping. 2017 IEEE International Conference on Communications (ICC), Paris, France, 2017, pp. 1–6, doi: 10.1109/ICC.2017.7997113.

[10] Tripathy, B. K.—Mitra, A.: An Algorithm to Achieve k-Anonymity and l-Diversity Anonymisation in Social Networks. 2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN), Sao Carlos, Brazil, 2012, pp. 126–131, doi: 10.1109/CASoN.2012.6412390.

[11] Sweeney, L.: Achieving k-Anonymity Privacy Protection Using Generalization and Suppression. International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems, Vol. 10, 2002, pp. 571–588, doi: 10.1142/S021848850200165X.

[12] Sweeney, L.: k-Anonymity: A Model for Protecting Privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, Vol. 10, 2002, No. 5, pp. 557–570, doi: 10.1142/S0218488502001648.

[13] Machanavajjhala, A.—Kifer, D.—Gehrke, J.—Venkitasubramaniam, M.: L-Diversity: Privacy Beyond k-Anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD), Vol. 1, March 2007, No. 1, pp. 1–52, doi: 10.1145/1217299.1217302.

[14] Samet, H.: Foundations of Multidimensional and Metric Data Structures. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2005.

[15] Fung, B. C. M.—Wang, K.—Yu, P. S.: Top-Down Specialization for Information and Privacy Preservation. 21st International Conference on Data Engineering (ICDE '05), Tokyo, Japan, 2005, pp. 205–216, doi: 10.1109/ICDE.2005.143.

[16] Fung, B. C. M.—Wang, K.—Chen, R.—Yu, P. S.: Privacy-Preserving Data Publishing: A Survey of Recent Developments. ACM Computing Surveys, Vol. 42, 2010, No. 4, Art. No. 14, pp. 1–53, doi: 10.1145/1749603.1749605.

[17] Liu, X.—Xie, Q.—Wang, L.: A Personalized Extended (a, k)-Anonymity Model. 2015 Third International Conference on Advanced Cloud and Big Data (CBD), Yangzhou, China, 2015, pp. 234–240, doi: 10.1109/CBD.2015.45.

[18] Guo, L.—Guo, S.—Wu, X.: Privacy Preserving Market Basket Data Analysis. In: Kok, J. N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (Eds.): Knowledge Discovery in Databases: PKDD 2007. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4702, 2007, pp. 103–114, doi: 10.1007/978-3-540-74976-9_13.

[19] Shen, Y.—Guo, G.—Wu, D.—Fan, Y.: A Novel Algorithm of Personalized-Granular k-Anonymity. Proceedings 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC), 2013, Shenyang, China, pp. 1860–1866, doi: 10.1109/MEC.2013.6885357.

[20] Meyerson, A.—Williams, R.: On the Complexity of Optimal k-Anonymity. Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '04), New York, USA, 2004, pp. 223–228, doi: 10.1145/1055558.1055591.

[21] Han, J.—Yu, H.—Yu, J.—Cen, T.: A Complete (α, k)-Anonymity Model for Sensitive Values Individuation Preservation. 2008 International Symposium on Electronic Commerce and Security, Guangzhou City, China, 2008, pp. 318–323, doi: 10.1109/ISECS.2008.92.

[22] Sei, Y.—Okumura, H.—Takenouchi, T.—Ohsuga, A.: Anonymization of Sensitive Quasi-Identifiers for l-Diversity and t-Closeness. IEEE Transactions on Dependable and Secure Computing, Vol. 16, 2019, No. 4, pp. 580–593, doi: 10.1109/TDSC.2017.2698472.

[23] Xiao, X.—Tao, Y.: Anatomy: Simple and Effective Privacy Preservation. Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB '06), Seoul, Korea, September 2006, pp. 139–150.

[24] Jin, X.—Zhang, M.—Zhang, N.—Das, G.: Versatile Publishing for Privacy Preservation. Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '10), 2010, pp. 353–362, doi: 10.1145/1835804.1835851.

[25] Tran, Q.—Sato, H.: A Solution for Privacy Protection in MapReduce. 2012 IEEE 36th Annual Computer Software and Applications Conference, Izmir, Turkey, 2012, pp. 515–520, doi: 10.1109/COMPSAC.2012.70.

[26] Jain, P.—Gyanchandani, M.—Khare, N.: Improved k-Anonymity Privacy-Preserving Algorithm Using Madhya Pradesh State Election Commission Big Data. In: Krishna, A., Srikantaiah, K., Naveena, C. (Eds.): Integrated Intelligent Computing, Communication, and Security. Springer, Singapore, Studies in Computational Intelligence, Vol. 771, 2019, pp. 1–10, doi: 10.1007/978-981-10-8797-4_1.

[27] Koli, A.—Shinde, S.: Parallel Decision Tree with Map Reduce Model for Big Data Analytics. International Conference on Trends in Electronics and Informatics (ICEI 2017), Tirunelveli, India, 2017, pp. 735–739, doi: 10.1109/ICOEI.2017.8300800.

[28] Mohamed, N.—Fung, B. C. M.—Hung, P. C. K.—Lee, C. K.: Centralized and Distributed Anonymization for High-Dimensional Healthcare Data. ACM Transactions on Knowledge Discovery from Data, Vol. 4, 2010, No. 4, Art. No. 18, doi: 10.1145/1857947.1857950.

[29] Jain, P.—Gyanchandani, M.—Khare, N.: Data Privacy for Big Data Publishing Using Newly Enhanced PASS Data Mining Mechanism. Book Chapter. In: Thomas, C. (Ed.): Data Mining. Intech Open Publisher, 2018, doi: 10.5772/intechopen.77033.

[30] Zakerzadeh, H.—Osborn, S. L.: FAANST: Fast Anonymizing Algorithm for Numerical Streaming DaTa. In: Garcia-Alfaro, J., Navarro-Arribas, G., Cavalli, A., Leneutre, J. (Eds.): Data Privacy Management and Autonomous Spontaneous Security (DPM 2010, SETOP 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6514, 2011, pp. 36–50, doi: 10.1007/978-3-642-19348-4_4.

[31] Guo, K.—Zhang, Q.: Fast Clustering-Based Anonymization Approaches with Time Constraints for Data Streams. Knowledge-Based Systems, Vol. 46, 2013, pp. 95–108, doi: 10.1016/j.knosys.2013.03.007.

[32] Mohammadian, E.—Noferesti, M.—Jalili, R.: FAST: Fast Anonymization of Big Data Streams. Proceedings of the 2014 International Conference on Big Data Science and Computing (BigDataScience '14), 2014, Art. No. 23, pp. 1–8, doi: 10.1145/2640087.2644149.

[33] Yadav, G. S.—Ojha, A.: A Fast and Efficient Data Hiding Scheme in Binary Images. 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Piraeus, Greece, 2012, pp. 79–84, doi: 10.1109/IIH-MSP.2012.25.

[34] LeFevre, K.—DeWitt, D. J.—Ramakrishnan, R.: Mondrian Multidimensional k-Anonymity. 22nd International Conference on Data Engineering (ICDE '06), Atlanta, GA, USA, IEEE, 2006, pp. 1–11, doi: 10.1109/ICDE.2006.101.

[35] Zakerzadeh, H.—Aggarwal, C. C.—Barker, K.: Privacy-Preserving Big Data Publishing. Proceedings of 27th International Conference on Scientific and Statistical Database Management (SSDBM '15), ACM, 2015, Art. No. 26, 11 pp., doi: 10.1145/2791347.2791380.

[36] Mehta, B. B.—Rao, U. P.: Privacy Preserving Big Data Publishing: A Scalable k-Anonymization Approach Using MapReduce. IET Software, Vol. 11, 2017, No. 5, pp. 271–276, doi: 10.1049/iet-sen.2016.0264.

[37] Cattral, R.—Oppacher, F.: Poker Hand Data Set. UCI Repository of Machine Learning Databases, University of California, School of Information and Computer Science, Irvine, CA, 2007.

[38] Kohavi, R.—Becker, B.: Adult Data Set. UCI Repository of Machine Learning Databases, University of California, School of Information and Computer Science, Irvine, CA, 1996.

Priyank Jain is working as Ph.D. Research Scholar. He has more than eight years of experience as Assistant Professor and in the research field. He has experience from Indian Institute of Management, Ahmedabad, India (IIM A) in the research field. His Ph.D. is in the big data privacy area. His educational qualification is M.Tech. and B. E. in information technology. His areas of specialization are big data, big data privacy and security, data mining, privacy-preserving, and information retrieval. He has publications in various international conferences, international journals and national conferences. He is a member of HIMSS.

Manasi Gyanchandani is working as Assistant Professor in MANIT Bhopal. She has more than 20 years of experience. She obtained her Ph.D. in computer science and engineering. Her area of specialization is in big data, big data privacy and security, data mining, privacy-preserving, artificial intelligence, expert system, neural networks, intrusion detection and information retrieval. She has publications in 8 international conferences, 15 international journals and 8 national conferences. She has her Life Time Membership of ISTE.

Nilay Khare is working as Professor in MANIT Bhopal. He has more than 21 years of experience. He obtained his Ph.D. in computer science and engineering. His areas of specialization are big data, big data privacy and security, wireless networks, theoretical computer science. His publications are in 54 international and national conferences, and international journals. He has his Life Time Membership of ISTE.

Computing and Informatics, Vol. 39, 2020, 568–586, doi: 10.31577/cai_2020_3_568

NEW SOFTWARE TOOL FOR MODELLING AND CONTROL OF DISCRETE-EVENT AND HYBRID SYSTEMS USING PETRI NETS

Erik Kučera, Oto Haffner, Peter Drahoš, Ján Cigánek

Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Bratislava, Slovakia
e-mail: {erik.kucera, oto.haffner}@stuba.sk

Juraj Štefanovič, Štefan Kozák

Faculty of Informatics
Pan-European University
Bratislava, Slovakia
e-mail: {juraj.stefanovic, stefan.kozak}@paneurouni.com

Abstract. The main aim of the proposed paper is to design a new software tool for modelling and control of discrete-event and hybrid systems using Arduino and similar microcontrollers. To accomplish these tasks a new tool called PN2ARDUINO based on Petri nets is proposed which is able to communicate with the microcontroller. Communication with the microcontroller is based on the modified Firmata protocol hence the control algorithm can be implemented on all microcontrollers that support this type of protocol. The developed software tool has been successfully verified in control of laboratory systems. It can also be used for education and research purposes as it offers a graphical environment for designing control algorithms for hybrid and mainly discrete-event systems. The proposed tool can improve education and practice in the field of cyber-physical systems (Industry 4.0).

Keywords: Discrete-event systems, hybrid systems, system control, microcontroller, Firmata protocol

Mathematics Subject Classification 2010: 93C65

1 INTRODUCTION

Development of various systems is a complex discipline that includes many activities, e.g. system design, specification of required properties, implementation, testing and further development of the system [1]. As these operations are challenging and important for the final product, it is appropriate and necessary to create a model of the system [2, 3]. Development of control methods for discrete-event and hybrid systems belongs to modern trends in automation and mechatronics. A hybrid system is a combination of continuous-time and discrete-event systems. Control of such systems brings new challenges due to the necessity to join control methods of discrete-event systems (where the Petri nets formalism can be helpful) and classic control methods of continuous-time systems [4]. With good methodology and software modules, these approaches can be synergistically combined yielding an appropriate and unique control system that allows harmonizing discrete-event and continuous-time control methods (e.g. PID algorithms). Effective cooperation of these approaches allows to control hybrid systems. This approach is useful in systems which require using different control algorithms (for example PID controllers with different parameters) according to the state of the system. The concept of Petri nets is capable to manage these control rules in a very efficient, robust and well-arranged (graphical) way. This paper presents new Petri Net tools for modelling and control of discrete-event and hybrid systems. Case studies dealing with control of a laboratory fire alarm system and a DC motor are included.

In papers [5] and [6] authors deal with usage of hybrid and colour Petri nets for modelling traffic on crossroads and on highways. From these authors, there are also interesting projects in the field of manufacturing systems [7] and [8]. Unfortunately, it is not mentioned whether the results are only theoretical models, or they were simulated using an SW tool or deployed in practice.

An interesting software tool named Visual Object Net++ that supports hybrid Petri nets was developed in [9]. There are many papers mainly from an author of [10] and [11] describing capabilities of Visual Object Net++. However, this tool is not open-source and has not been further developed.

The SW tool Snoopy [12] offers modelling based on many Petri nets classes like stochastic, hybrid, colour, music Petri nets, etc. Using this tool, many types of research in biology and chemistry are being solved. Unfortunately, the source code is not available.

Coloured Petri nets are used for modelling of automated storage and retrieval system in [13] and [14].

One of interesting research approaches is the Modelica language and the OpenModelica open-source tool. There is a library that supports modelling by Petri nets in this tool. One of the advantages of OpenModelica is that a PN model can be connected with other Modelica components. The first Petri net toolbox was introduced in [15], its extension is described in [16]; an important addition (called PNlib) including support of extended hybrid Petri nets for modelling of processes in biological organisms is described in [17] and [18]. However, this tool was devel-

oped primarily for the commercial tool Dymola and not for OpenModelica, so its extensibility and applicability in scientific research are limited. In 2015, the team that developed PNlib published a modified version of PNlib that partially worked in OpenModelica. Unfortunately, it was not possible to use OpenModelica for control purposes using microcontrollers because the COM port communication support was missing.

The above survey has shown there is a lack of tools based on Petri net formalism that support control of real systems. As a result, it was necessary to develop an original solution for control of discrete and hybrid systems based on microcontrollers using Petri nets formalism.

2 DESCRIPTION OF DEVELOPED SW TOOL PN2ARDUINO

As a basis for the newly-developed SW tool, the PNEditor was chosen [19]. This tool is open-source. The developed extension of this tool is called PN2ARDUINO and was fully tested in [20] and [21]. The main topic of this paper includes an introduction to this developed software that can be used for control of discrete-event and hybrid systems, and its verification on laboratory discrete-event and hybrid systems.

Petri Net Logic in PC	**Petri Net Logic in Microcontroller**
limited capability of real-time control	real-time control
much more computation and memory resources available	limited computation and memory resources
code in microcontroller does not need recompiling	during development repeated compiling is needed
PC must be still online	independence of control unit

Table 1. Comparison of two concepts of system control using Petri nets

There are several Petri net-based control concepts. Petri net as a control logic has to be connected with the controlled system (e.g. using a microcontroller). One of the main aspects of the control system design is the question whether the Petri net's logic should be stored in the microcontroller or in the PC able to communicate with the microcontroller. Both approaches have both advantages and disadvantages.

If the Petri net's logic is stored in the microcontroller, the main advantage is the control unit independence from the software application (program on a PC). The Petri net logic is modelled using a PC, and then the Petri net is translated into a program code which is loaded into the microcontroller. Afterwards, the PC and the microcontroller can be disconnected. Another advantage is the capability of real-time control. Disadvantages include limited computational and memory resources of the microcontroller, need for repeated program compiling and its uploading into the microcontroller (mainly during the development phase). The proposed solution is shown in Figure 1.

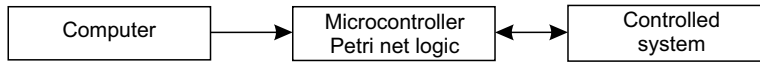


Figure 1. Basic scheme of proposed solution – Petri net's logic in microcontroller

When the Petri net's control logic is stored on a PC in a specialized SW application, it is possible to control the system directly. In the microcontroller, only the program with communication protocol is stored. This communication protocol (in our case Firmata [22]) is used for communication between the PC and the microcontroller. This solution eliminates the necessity of recompiling and reuploading the program during the development. The next advantage is the elimination of restrictions on computing and storage resources because a PC has almost unlimited resources compared with a microcontroller. One of the disadvantages is that the control system cannot respond in real time. The proposed solution is shown in Figure 2. These differences are specified in Table 1.

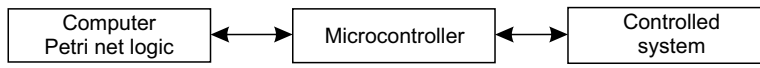


Figure 2. Basic scheme of proposed solution – Petri net's logic in PC

New software module PN2ARDUINO is based on the second approach. The Petri net runs on a personal computer. For communication between SW application and microcontroller, the Firmata protocol [22] has been used. Firmata is a protocol designed for communication between a microcontroller and a computer (or a mobile device like smartphone, tablet, etc.). It is based on MIDI messages [23]. This protocol can be implemented in firmware of various microcontrollers; mostly Arduino-family microcontrollers are used. On the PC, a client library is needed. These libraries are available for many languages like Java, Python, .NET, PHP, etc.

On the Arduino side, the Standard Firmata 2.3.2 version is used, the client application on the PC is based on Firmata4j 2.3.3 library which is programmed in Java. The advantage of using Firmata consists in the possibility of using another microcontroller compatible with Firmata.

PN2ARDUINO extends PNEditor with many features. For Petri nets modelling, there is a possibility of adding time delays to transitions, and capacity for places. Also, the automatic mode of transitions firing was added for automatic control purposes as the only manual mode is available in PNEditor.

In PN2ARDUINO, a new module is added to PNEditor to enable communication with the compatible microcontroller. This module consists of two parts. The first part establishes connection with the microcontroller by setting the COM port where the microcontroller is connected. The second part provides a capability of adding Arduino components to Petri net places and transitions. The following

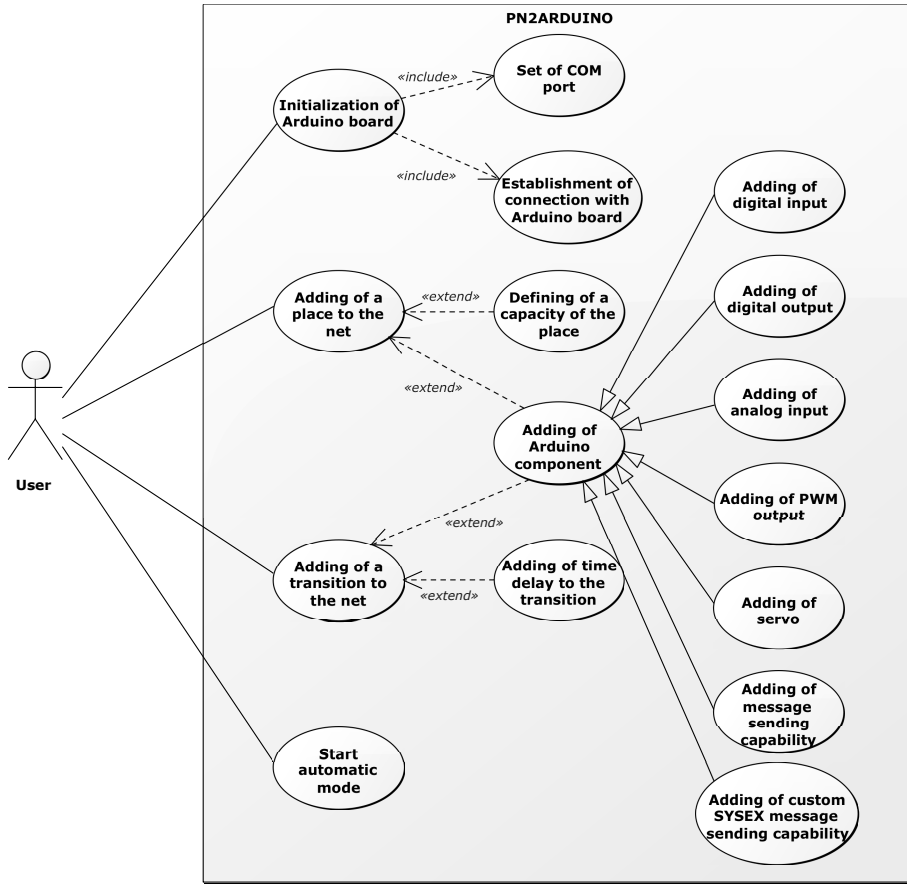


Figure 3. PN2ARDUINO – use-case diagram

types of Arduino components are supported: digital input and output, analog input, servo control, PWM output, message sending, and custom SYSEX message [22] sending.

The use-case diagram of the developed SW tool is depicted in Figure 3, and the class diagram is shown in Figure 4.

As it was stated, transitions and places can be associated with Arduino components. Digital and analog inputs serve as enabling conditions for Petri net transitions. Digital and PWM outputs and messages are used as executors of the respective actions.

The interesting functionality is the capability of sending custom SYSEX messages. The user has to enter a SYSEX command (`0x00 - 0x0F`) and optionally also the content of the message. The message is sent when the token comes to the place or when the transition is fired. SYSEX messages have been used e.g. in the

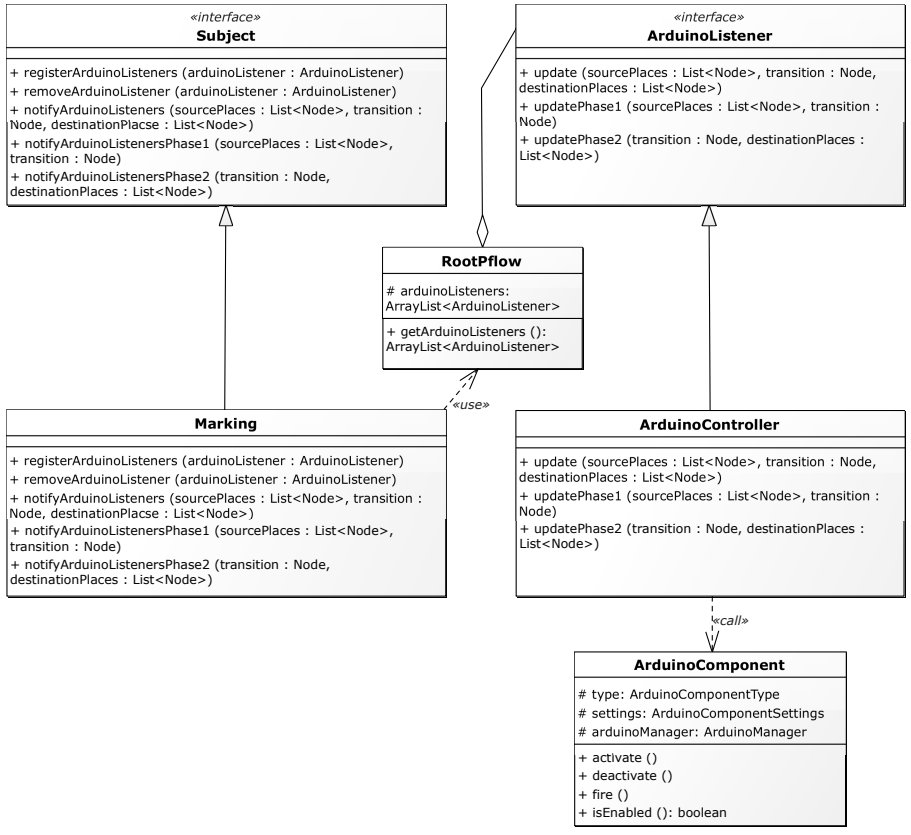


Figure 4. PN2ARDUINO – class diagram

proposed hybrid control example in the last section of this article. Here, the SYSEX message notifies the microcontroller that a different PID control algorithm is to be used. Then the PID algorithm is switched, and the controlled system remains stable.

A main window of PN2ARDUINO consists of a quick menu, main menu, canvas for Petri net modelling and log console. PN2ARDUINO supports two modes – a design mode and a control mode, the control mode can be manual or automatic.

Firstly, it is necessary to initialize communication with Arduino (`Setup board` in the menu). Then it is possible to add Arduino component to the place or to the transition (Figure 5). The example of analog input is shown in Figure 6.

Time politics are also supported – it is possible to add time delay to the transitions which can be deterministic or stochastic.

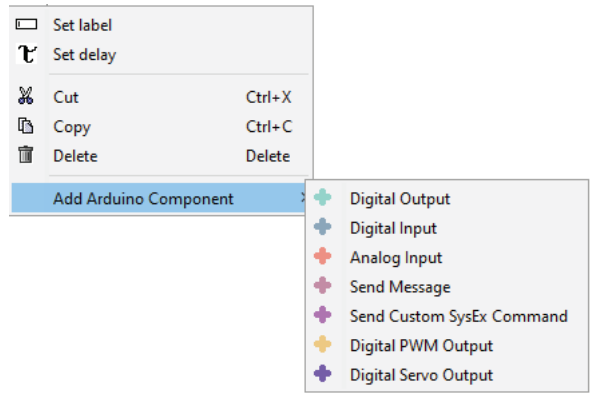


Figure 5. PN2ARDUINO – adding of Arduino component

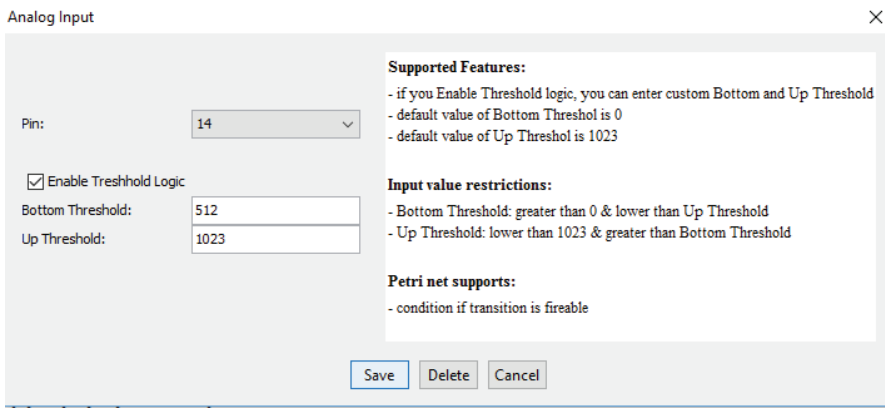


Figure 6. PN2ARDUINO – analog input

3 CASE STUDY: CONTROL OF LABORATORY DISCRETE-EVENT SYSTEM

For verification of the developed software tool and discrete-event systems control method it was necessary to design a laboratory model of such a system. A fire alarm model was built. The scheme can be seen in Figure 7.

The fire alarm model consists of an active buzzer, photo-resistor, three resistors and an NPN transistor. The NPN transistor is mandatory for active buzzer connection. The LED of Arduino in pin 13 is also used. A photo-resistor was used instead of a smoke sensor to simplify the experiment.

Next, the behaviour of the system has to be defined. When the photoresistor detects an excessive lighting (experimentally determined as an input value greater

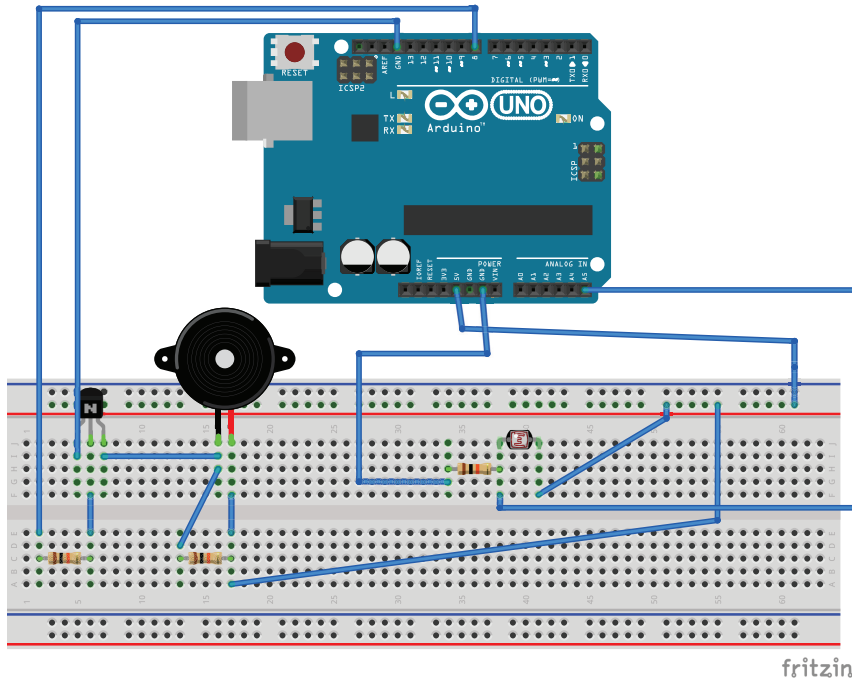


Figure 7. The scheme of laboratory model of fire alarm

than 799 on the analog pin of Arduino Uno which resolution is from 0 to 1 023) the intermittent tone of the buzzer is turned on. This tone alternates with LED lighting. When the value on the analog pin drops below 800, the sound and light effects stop. This is repeated cyclically.

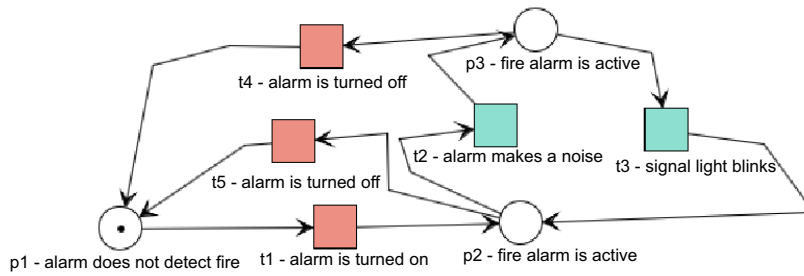


Figure 8. PN for fire alarm (initial marking)

Initial marking of modelled timed Petri Net interpreted for control (or sometimes called as interpreted timed Petri net) in PN2ARDUINO is shown in Figure 8.

Places of the Petri net (Figure 8 – Figure 10) correspond to the following states:

- p_1 – alarm does not detect fire,
- p_2 and p_3 – alarm is active (fire was detected).

Transitions of Petri net (Figure 8 – Figure 10) correspond with the following actions/events:

- t_1 – alarm is turned on,
- t_2 – alarm makes a noise,
- t_3 – signal light blinks,
- t_4 and t_5 – alarm is turned off.

The token in place p_1 corresponds to the state when the fire alarm is not activated because the photo-resistor has not detected the light intensity threshold.

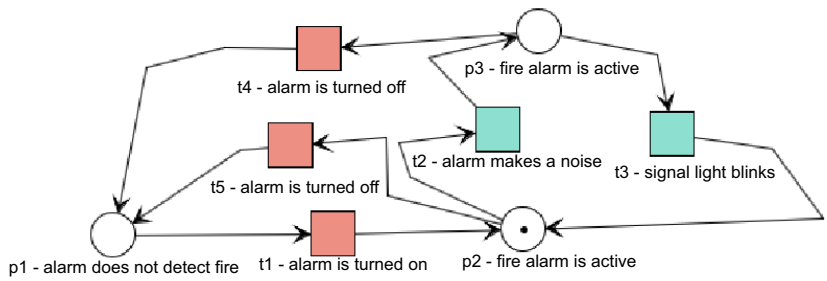


Figure 9. PN for fire alarm (t_1 is fired)

When a value greater than 799 is detected on the analog pin of Arduino, the transition t_1 is fired. This transition is associated with Arduino component *Analog Input* where the range of input values is set. The transition is enabled depending on this range.

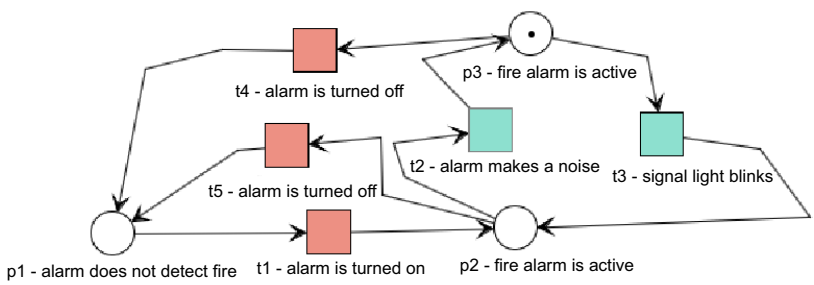


Figure 10. PN for fire alarm (t_2 is fired)

In Figure 9, the token is in the place p_2. Transition t_2 is associated with Arduino component *Digital Output* (pin 8 in this case) where the buzzer is connected. This transition has also associated the function of a time delay (2 seconds) which means that the transition firing (and buzzer sound effect) lasts for 2 seconds.

In Figure 10, the token is in the place p_3 (Figure 10). Transition t_3 is associated with Arduino component *Digital Output* (pin 13 in this case) where the build-in LED is connected. The time delay is set to 1 second, i.e. the LED diode is turned on for 1 second.

This process is repeated cyclically, and it stops when the value on the analog pin drops under 800. Then the transition t_4 or t_5 is fired and the token moves to the place p_1 when the fire alarm does not detect the fire.

We can conclude that the ability of discrete-event control with PN2ARDUINO was successfully verified and can be generalized for other applications.

4 CASE STUDY: CONTROL OF LABORATORY HYBRID SYSTEM

To verify the proposed software tool for hybrid systems control it was necessary to find an appropriate laboratory model. A DC motor with encoder was chosen; its parameters are in Table 2. An incremental encoder is used to measure speed for the feedback. The measured speed of the DC is the process value.

Actuators Conditions	
Rated voltage	6.0 V (DC)
Humidity range	0 %–90 %
Temperature range	$-20\,^{\circ}\mathrm{C} \sim +60\,^{\circ}\mathrm{C}$
No-Load Characteristics	
No-load current	≤ 200 mA
No-load speed	$185 \pm 10\,\%$ rpm
Load Characteristics	
Rated load	0.0883 N.m
Rated current	≤ 550 mA
Rated speed	$135 \pm 10\,\%$ rpm
Starting torque	0.4413 N.m
Locked-rotor current	≥ 2.0 A

Table 2. Specification of DC motor

The DC motor was connected to Arduino Uno using a motor shield module based on dual full bridge driver L298. Using the motor shield, it is possible to independently control speed and direction of rotation of the DC motor. For speed measurement the hardware interruptions functionality of Arduino Uno has been used.

The speed of the motor is set by a pin denoted "PWM A". When the input is set to "PWM = 255" the Arduino program shows 186 rpm which approximately corresponds with parameters as stated by the manufacturer.

The next step was measurement of the steady state I/O characteristics. The input (armature) voltage ranges from 0 V to 5 V which corresponds to a PWM signal from 0 to 255 (8-bit resolution), the sampling period was chosen – 0.05 s.

The resulting steady state I/O characteristics is in Figure 11.

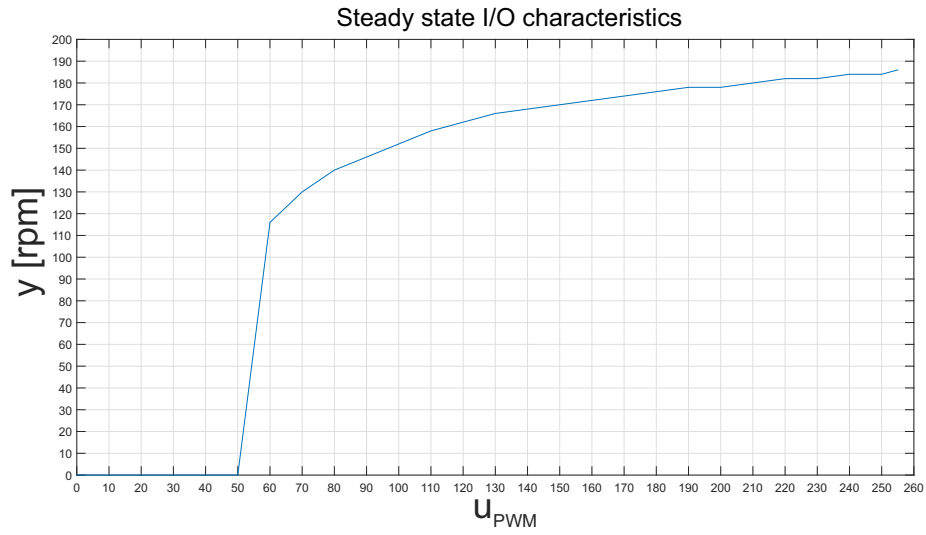


Figure 11. Steady state I/O characteristics of DC motor

To be able to use linear dynamic models, the working points had to be chosen from linear parts of the I/O characteristics. Two working points have been chosen (u_{P_1},y_{P_1}) and (u_{P_2},y_{P_2}) where:

$$u_{P_1} = 80 \rightarrow y_{P_1} = 140\,\text{rpm}, \tag{1}$$

$$u_{P_2} = 170 \rightarrow y_{P_2} = 174\,\text{rpm}. \tag{2}$$

Since the controller has been designed for a real system (fast dynamics, large noise, uncertainties), the controller parameters were tuned using practice-oriented design methods. The designed PID controller was implemented using the Arduino PID Library [24]. Creation of a PID class object has the following syntax:

- `PID (&Input, &Output, &Setpoint, Kp, Ki, Kd, Direction)`
  - `Input`: controlled variable (`double`), rpm of the motor
  - `Output`: control variable (`double`), in this case input voltage – PWM (0–255)
  - `Setpoint`: setpoint (`double`), desired rpm of the motor
  - `Kp, Ki, Kd`: tuning parameters (`double >= 0`)

– `Direction`: Either DIRECT or REVERSE – determines which direction the output will move when faced with a given error.

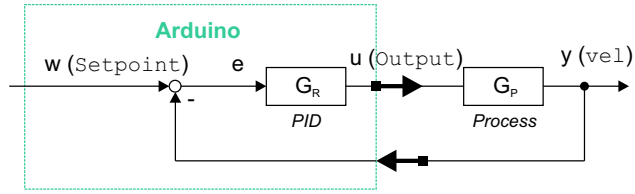


Figure 12. Block diagram of PID controller in a feedback loop

The closed-loop scheme is in Figure 12. The monotype font was used for variables names in the Arduino program. The `Setpoint` is the desired speed (rpm). Due to the used data type in the Arduino program (`long`), multiples of ten of the setpoint are used. Using an extra order enables to deal with an equivalent of a number with one decimal place. `Output` is the control variable ranging between 0 and 255 (8 bits) corresponding to the input voltage between 0 V and 5 V (PWM 0–255).

Figure 13. Step response (closed loop) – 2nd working point

Experiments showed that designing an effective PID controller for higher speeds (2nd working point: $\omega = 176$ rpm) is not complicated. A satisfactory control performance can be achieved by tuning individual PID controller parameters.

$$G_R = P + \frac{I}{s} + Ds. \tag{3}$$

However, it was not so easy to design an effective controller in the 1st working point ($\omega = 140$ rpm); mostly, the closed-loop system was not stable. Hence, lower P and I values had to be used. It was expected that this controller will be effective also in the 2nd working point however with a worse performance (too long settling time). To switch between different control algorithms in individual working points, the proposed software for control of hybrid systems using Petri Nets can be used.

For the 2nd working point, a PID controller with parameter values $P = 0.83$; $I = 5$; $D = 0.005$ was designed. The closed loop step response is shown in Figure 13 whereby a PWM step from 176 to 186 was realized in $t = 5$ s. The settling time is 1.1 s (considering a tolerance ± 2 rpm). This controller was tested also in the 1st working point (for a step from 140 rpm to 146 rpm). It is obvious from the corresponding step response in Figure 14 that this controller does not provide a satisfactory performance.

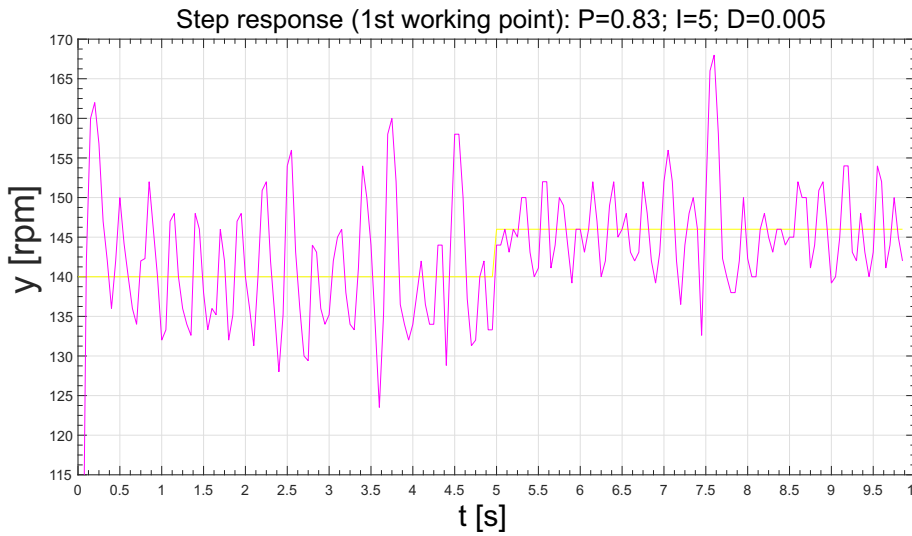


Figure 14. Step response (closed loop) – 1st working point – with inappropriate controller

Hence, for the 1st working point a different PID controller was designed with $P = 0.0001$; $I = 1$; $D = 0.01$. The closed loop step response in Figure 15 shows that the controller works properly (the settling time is 1.3 s). When applied in the 2nd working point, this controller again did not work properly, as expected (larger settling time achieved with a PID controller with smaller P and I). The achieved performance evaluated from the step response in Figure 16 is worse compared with the 1st controller ($P = 0.83$; $I = 5$; $D = 0.005$), the settling time 2.75 s is much larger than in case of the first controller (1.1 s).

The analysis of the achieved results revealed the necessity to use different controllers in individual working points, or to design the controller using some ad-

Figure 15. Step response (closed loop) – 1st working point – under PID controller

vanced control design approach (robust, gain scheduled, switched). In case of switching between multiple controllers according to the working point it is possible to use the developed software module PN2ARDUINO. Switching between controllers and setpoints is based on SYSEX messages. Arduino and other microcontrollers that support Firmata protocol can be used. Development and verification of this software module are the most interesting results of the presented research.

A demonstration example of the proposed control method is in Figure 17. Consider the above-mentioned DC motor which has to operate in two modes (working points). For effective setpoint tracking by the DC motor speed, controllers with different parameters have to be used (a different controller for each mode). Switching between individual working points is carried out using a potentiometer connected to the analog input of the Arduino Uno microcontroller. Switching between controllers is provided by transitions *switch1* and *switch2* of Petri net according to the input value from the potentiometer. Input from the analog pin in Arduino is represented by a value ranging between 0 and 1023. The mean value (512) was used as a threshold. In the moment when the token in Petri net is moved to the places *setpoint1* or *setpoint2*, a SYSEX message is sent. This message ensures the execution of a user-defined program code on the Arduino side, in this case the control algorithm. The (PID) algorithm for continuous-time control is independent of Firmata messaging, so it provides real-time control. The provided hybrid systems control case study is a basic example. Researchers in hybrid control design can use it for different and even more complicated scenarios.

Figure 16. Step response (closed loop) – 2nd working point – under inappropriate controller

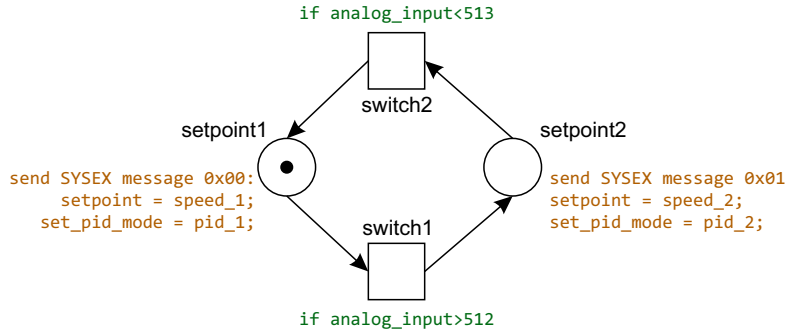


Figure 17. Control scheme for hybrid system using PN2ARDUINO

5 CONCLUSIONS

The paper presents a new software tool PN2ARDUINO extending the PNEditor to communicate with microcontrollers that support the Firmata protocol. This allows controlling discrete-event and hybrid systems using timed interpreted Petri nets with the developed software tool. The developed SW tool supports the control paradigm when in the microcontroller only the communication protocol is implemented. Petri nets control logic is stored in the computer which communicates with the microcontroller and sends control comands. The main advantage of the developed SW tool is a possibility to control complex discrete-event and hybrid systems using the

benefits of Petri nets formalism which can support many challenging scenarios. The next research will focus on the concept of Petri nets based control with the control logic directly implemented in the microcontroller.

Acknowledgment

This work has been supported by the Cultural and Educational Grant Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic, KEGA 038STU-4/2018 and KEGA 016STU-4/2020, and by the Slovak Research and Development Agency APVV-17-0190.

REFERENCES

[1] Kharitonov, D.—Tarasov, G.—Golenkov, E.: Modeling of Object-Oriented Programs with Petri Net Structured Objects. Computing and Informatics, Vol. 36, 2017, No. 5, pp. 1063–1087, doi: 10.4149/cai_2017_5_1063.

[2] Babič, M.—Hluchý, L.—Krammer, P.—Matovič, B.—Kumar, R.—Kovač, P.: New Method for Constructing a Visibility Graph-Network in 3D Space and a New Hybrid System of Modeling. Computing and Informatics, Vol. 36, 2017, No. 5, pp. 1107–1126, doi: 10.4149/cai_2017_5_1107.

[3] Julia, S.—do Nascimento Vale, L.—Soares Passos, L. M.: Functional Testing Using Object WorkFlow Nets. Computing and Informatics, Vol. 35, 2016, No. 3, pp. 719–743.

[4] Szymczak, A.—Paszyński, M.—Pardo, D.—Paszyńska, A.: Petri Nets Modeling of Dead-End Refinement Problems in a 3D Anisotropic *hp*-Adaptive Finite Element Method. Computing and Informatics, Vol. 34, 2015, No. 2, pp. 425–457.

[5] Dotoli, M.—Fanti, M. P.—Iacobellis, G.: A Freeway Traffic Control Model by First Order Hybrid Petri Nets. 2011 IEEE Conference on Automation Science and Engineering (CASE), 2011, pp. 425–431, doi: 10.1109/CASE.2011.6042526.

[6] Fanti, M. P.—Iacobellis, G.—Mangini, A. M.—Ukovich, W.: Freeway Traffic Modeling and Control in a First-Order Hybrid Petri Net Framework. IEEE Transactions on Automation Science and Engineering, Vol. 11, 2014, No. 1, pp. 90–102, doi: 10.1109/TASE.2013.2253606.

[7] Dotoli, M.—Fanti, M. P.—Mangini, A. M.: Fault Monitoring of Automated Manufacturing Systems by First Order Hybrid Petri Nets. IEEE International Conference on Automation Science and Engineering (CASE 2008), 2008, pp. 181–186, doi: 10.1109/COASE.2008.4626493.

[8] Costantino, N.—Dotoli, M.—Falagario, M.—Fanti, M. P.—Mangini, A. M.: A Model for Supply Management of Agile Manufacturing Supply Chains. International Journal of Production Economics, Vol. 135, 2012, No. 1, pp. 451–457, doi: 10.1016/j.ijpe.2011.08.021.

[9] Matsuno, H.—Doi, A.—Drath, R.—Miyano, S.: Genomic Object Net: Object Oriented Representation of Biological Systems. Genome Informatics, Vol. 11, 2000, pp. 229–230.

[10] Drighiciu, M. A.—Manolea, G.: Application des Reseaux de Petri Hybrides a l'Etude des Systemes de Production a Haute Cadence. 2010 (in French).

[11] Drighiciu, M.—Cismaru, D.: Modeling a Water Bottling Line Using Petri Nets. Annals of the University of Craiova, Electrical Engineering Series, 2013, No. 37, pp. 110–115.

[12] Rohr, C.—Marwan, W.—Heiner, M.: Snoopy – A Unifying Petri Net Framework to Investigate Biomolecular Networks. Bioinformatics, Vol. 26, 2010, No. 7, pp. 974–975, doi: 10.1093/bioinformatics/btq050.

[13] Kučera, E.—Nižnanská, M.—Kozák, Š.: Advanced Techniques for Modelling of AS/RS Systems in Automotive Industry Using High-Level Petri Nets. 2015 16th International Carpathian Control Conference (ICCC), IEEE, 2015, pp. 261–266, doi: 10.1109/CarpathianCC.2015.7145085.

[14] Kučera, E.—Haffner, O.—Kozák, Š.: Modelling and Control of AS/RS Using Coloured Petri Nets. 2016 Cybernetics & Informatics (K & I), IEEE, 2016, pp. 1–6, doi: 10.1109/CYBERI.2016.7438532.

[15] Mosterman, P. J.—Otter, M.—Elmqvist, H.: Modeling Petri Nets as Local Constraint Equations for Hybrid Systems Using Modelica. Available at: `https://www.modelica.org/publications/papers/scsc98fp.pdf`, 1998.

[16] Fabricius, S. M. O.—Badreddin, E.: Modelica Library for Hybrid Simulation of Mass Flow in Process Plants. Proceedings of the 2nd International Modelica Conference, Oberpfaffenhofen, Germany, Citeseer, 2002, pp. 225–234.

[17] Pross, S.—Bachmann, B.: A Petri Net Library for Modeling Hybrid Systems in OpenModelica. Proceedings 7th Modelica Conference, Como, Italy, 2009, pp. 454–462, doi: 10.3384/ecp09430014.

[18] Pross, S.—Bachmann, B.: PNlib – An Advanced Petri Net Library for Hybrid Process Modeling. Proceedings of the 9th International Modelica Conference, Munich, Germany, 2012, pp. 47–56, doi: 10.3384/ecp1207647.

[19] Riesz, M.—Sečkár, M.—Juhás, G.: PetriFlow: A Petri Net Based Framework for Modelling and Control of Workflow Processes. In: Donatelli, S., Kleijn, J., Machado, R. J., Fernandes, J. M. (Eds.): Recent Advances in Petri Nets and Concurrency. CEUR Workshop Proceedings, Vol. 827, 2012, pp. 191–205.

[20] Češeková, A.: Control of Laboratory Discrete Event Systems. Master's thesis, Slovak University of Technology in Bratislava, 2016 (in Slovak).

[21] Kučera, E.: Modelling and Control of Hybrid Systems Using High-Level Petri Nets. Ph.D. Dissertation, Slovak University of Technology in Bratislava, 2016 (in Slovak).

[22] Steiner, H.: Firmata: Towards Making Microcontrollers Act Like Extensions of the Computer. Proceedings of the International Conference on New Interfaces for Musical Expression (NIME), 2009, pp. 125–130, doi: 10.5281/zenodo.1177689.

[23] MIDI Association: Summary of Midi Messages. Available at: `https://www.midi.org/specifications/item/table-1-summary-of-midi-message`, 2016.

[24] Comnes, B.—La Rosa, A.: Arduino PID Example Lab. Portland State University, 2013.

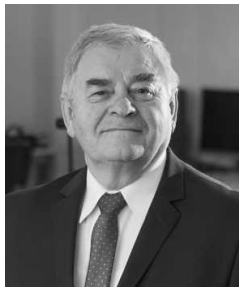
Erik Kučera graduated from the Slovak University of Technology, Faculty of Electrical Engineering (FEI STU) in Bratislava in 2013 and obtained his Ph.D. in mechatronic systems in 2016. He works at the Institute of Automotive Mechatronics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia. His focus is mainly on modern information and communication technologies and their use in the context of fourth industrial revolution Industry 4.0. This includes e.g. internet of things, virtual and mixed reality, cloud computing, new microcontrollers, etc.

Oto Haffner graduated from the Slovak University of Technology, Faculty of Electrical Engineering (FEI STU) in Bratislava in 2013 and obtained his Ph.D. in mechatronic systems in 2016. He works at the Institute of Automotive Mechatronics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovakia. His focus is mainly on modern machine vision methods and their use in the context of fourth industrial revolution Industry 4.0. This includes e.g. artificial intelligence, deep learning, etc.

Peter Drahoš graduated from the Slovak University of Technology, Faculty of Electrical Engineering (FEI STU) in Bratislava in 1985 and obtained his Ph.D. in automation and control in 2003. His main research interests include smart material actuators, sensors and automatic control. He is also interested in industrial communication systems. Since 2012, he is with the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, holding now the position of Associate Professor.

Ján Cigánek received his diploma and Ph.D. degree in automatic control from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology (FEI STU) in Bratislava, in 2005 and 2010, respectively. He is now Assistant Professors at the Institute of Automotive Mechatronics FEI STU in Bratislava. His research interests include optimization, robust control design, computational tools, and hybrid systems.

Juraj Štefanovič graduated from the Faculty of Electrical Engineering of the Slovak University of Technology in Bratislava, Slovakia in 1987 in microelectronic engineering, and received his Ph.D. in applied informatics from the Slovak University of Technology in 2000. He was a researcher at the Slovak Academy of Sciences until 1992. Since 1992 he was with the Slovak University of Technology in Bratislava, the Faculty of Electrical Engineering/Faculty of Informatics and Information Technologies at the Department of Informatics until 2014. Currently he is with the Institute of Applied Informatics at the Faculty of Informatics, Pan-European University in Bratislava as Vice-Dean for Research and International Relations. He is also the Executive Editor of International Journal of Information Technology Applications (ITA) and he is a manager of regional competition of First Lego League (robotic competition for youth teams) in Bratislava-Petržalka. His research interests include modelling and simulation – discrete dynamic models and complex models, his teaching activity covers also the field of operating systems and design for usability.



Štefan Kozák obtained his M.Sc. from the Slovak University of Technology in Bratislava in 1970 and his Ph.D. in technical cybernetics from the Slovak Academy of Sciences in 1978. He worked at the Institute of Technical Cybernetics, Slovak Academy of Sciences, in the field of control algorithms design and was Leader of a Research Team at the Institute of Applied Cybernetics in Bratislava. Since 1984 he was with the Department of Automatic Control Systems at the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava. Currently he is with the Institute of Automotive Mechatronics, Slovak University of Technology in Bratislava. His research interests include system theory, linear and nonlinear control methods, numerical methods and software for modeling, control, signal processing and embedded intelligent systems. He published more than 220 research papers in conference proceedings and international journals, and he organized several IFAC events held in Slovakia.

Computing and Informatics, Vol. 39, 2020, 587–616, doi: 10.31577/cai_2020_3_587

PSVDAG: COMPACT VOXELIZED REPRESENTATION OF 3D SCENES USING POINTERLESS SPARSE VOXEL DIRECTED ACYCLIC GRAPHS

Liberios Vokorokos, Branislav Madoš, Zuzana Bilanová

Department of Computers and Informatics
Faculty of Electrical Engineering and Informatics
Technical University of Košice
Letná 9, 042 00 Košice, Slovak Republic
e-mail: {liberios.vokorokos, branislav.mados, zuzana.bilanova}@tuke.sk

Abstract. This paper deals with the issue of geometry representation of voxelized three-dimensional scenes using hierarchical data structures. These include pointerless Sparse Voxel Octrees that have no pointers on child nodes and allow a compact binary representation. However, if necessary, there is a possibility to reconstruct these pointers for rapid traversing. Sparse Voxel Directed Acyclic Graphs added 32-bit pointers to child nodes and merging of common subtrees, which can be considered lossless compression. By merging common subtrees, no decompression overhead occurs at the time of traversing. The hierarchical data structure proposed herein – the Pointerless Sparse Voxel Directed Acyclic Graph – incorporates the benefits of both – pointerless Sparse Voxel Octrees (by avoiding storing pointers on child nodes) and Sparse Voxel Directed Acyclic Graphs (by allowing the merging of common subtrees due the introduction of labels and callers). The proposed data structure supports the quick and easy reconstruction of pointers by introducing the Active Child Node Count. It also potentially allows Child Node Mask compression of its nodes. This paper presents the proposed data structure and its binary-level encoding in detail. It compares the effectiveness of the representation of voxelized three-dimensional scenes (originally represented in OBJ format) in the proposed data structure with the data structures mentioned above. It also summarizes statistical data providing a more detailed description of the various parameters of the data structure for different scenes stored in multiple resolutions.

Keywords: Lossless data compression, sparse voxel octrees, SVO, sparse voxel directed acyclic graph, SVDAG, pointerless sparse voxel directed acyclic graph, PSVDAG, symmetry-aware sparse voxel directed acyclic graph, SSVDAG

Mathematics Subject Classification 2010: 68-P30

1 INTRODUCTION

The initial motivation for volumetric data representation was the need to process and visualize three-dimensional scientific, medical, and industrial data. Volume datasets are mostly spatially uniform, regular grids of scalar or vector values. They may be obtained through the acquisition using specialized appliances that fit for this purpose or represent the results of simulations or calculations. Computed Tomography (CT), MicroCT, or Magnetic Resonance Imaging (MRI) are devices suitable for data acquisition in medical applications. As far as the industries are concerned, Industrial Computed Tomography can be a source of data for assembly inspection and measurements, flaw detection, failure analysis, or reverse engineering. Voxelized polygonal three-dimensional models may act as another, often used, source of volume data.

As the development of volumetric data visualization continued (including the development of hardware to support it), its employment has also shifted to computer graphics, visual arts, and computer-based entertainment, including computer games, augmented or virtual reality. Volume rendering is also widely used in the movie industry. Notable examples of volumetric graphic visualization include the motion pictures XXX, Lord of the Rings, The Day After Tomorrow, Pirates of the Caribbean, or The Mummy [1]. The advantages of voxelized graphics include the possibility of implementing effects such as smoke, fog, snow, fire, and more.

Complex voxelized three-dimensional scenes, even taking only the geometry of the scene into account, can represent a significant amount of data. An example may be the geometry of a scene having a resolution of $64\,K^3$ ($65\,536 \times 65\,536 \times 65\,536$) voxels. Using a regular three-dimensional grid of 1 b per voxel for its representation, up to 32 TB of space is required to store it in an uncompressed form. Processing this type of data in its uncompressed form may be inefficient. If the goal is the real-time or interactive processing, requiring the storage of the entire amount of data in the computer's memory or the graphics card, this task may not be feasible. Especially when reasonable hardware resources and off-the-shelf hardware solutions are required. That is why significant effort is devoted to developing more compact representations of voxelized three-dimensional data (with lossless compression).

In this context, various data structures have emerged over the decades of development. Hierarchical data structures based on octant trees have gained considerable popularity. Among other advantages, they allow the efficient work with empty subspaces or an implicit Level of Detail (LOD) mechanism. A more comprehensive overview of GPU-friendly volume data representation and rendering is available in [2].

Sparse Voxel Octrees (SVO) are used as a hierarchical data structure to represent the geometry of voxelized three-dimensional scenes. SVO allows carving out empty suboctants, i.e., those containing no active voxels.

In 2013, with the emergence of Sparse Voxel Directed Acyclic Graphs (SVDAG), GPU friendly directed acyclic graphs with Common Subtree Merging (CSM) were

introduced. In this concept, a fully developed instance of the particular subtree, referenced (multiple times) by child node pointers, replaces two or more identical subtrees. CSM can be considered a lossless compression of the hierarchical data structure. In 2016, a GPU friendly Symmetry-aware Sparse Voxel Directed Acyclic Graphs (SSVDAG) emerged. SSVDAG added the CSM option, provided that transformations of these subtrees – namely reflective symmetry transformation – are used. SSVDAG allows further modification of the data structure – clustering leaf nodes, to form 4^3 grids of voxels – for further increase of the compression ratio.

On the other hand, in this context, it is necessary to observe how this data compaction manifests itself in the use of the data structure. The decompression overhead must be as small as possible. The reason is the need for on-the-fly decompression during traversing. This issue may also be considered a trade-off between the size of the binary representation of the data and the time it takes to perform operations. Data optimization performs data compaction while preserves the previous service functionality [3].

Modifying the presence, structure, and length of the binary representation of pointers to the child nodes is another factor that can reduce the size of the data structures mentioned above. An example is the SSVDAG data structure having 0, 16, and 32-bit pointers, which is a refinement over the previous SVDAG data structure, only having 0 and 32-bit pointers. On the other hand, pointerless SVOs have no pointers at all.

In this work, the authors attempted to create a hierarchical data structure without any pointers on child nodes. Therefore it will be significantly more compact than SVDAG and, considering the way of encoding information about the decomposition of individual suboctants, approaching pointerless SVO. On the other hand, like SVDAG, it will enable compaction of the data structure via the CSM, even without the presence of child node pointers, using labels and callers with a compact variable length of their binary representation, introduced in this paper. Frequency-based compaction will promote further compacting of labels and callers. Additional information facilitates the reconstruction of child node pointers into a form allowing quick and efficient traversal.

In more modern hierarchical data structures, the number of mask bits of potential child nodes increases from 1 b to 2 b, increasing the length of the Child Node Mask from 8 b to 16 b. Therefore a further goal was to investigate the possibility of compacting the Child Node Mask (CHNM) of the respective nodes. Pointers optimization offers the potential to rationalize the binary representation of this part of the hierarchical data structure node.

The contribution hereof lies in the following:

- the design of a Pointerless Sparse Voxel Directed Acyclic Graph (PSVDAG) hierarchical data structure allowing a compact pointerless representation of the geometry of three-dimensional scenes;
- the implementation of common subtrees merge with the absence of child node pointers at the same time. Introduction of the concept of labels and callers that

are replacing particular pointers of the hierarchical data structure, is enabling the CSM;

- the length optimization of the binary representation of labels and callers, using variable-length and frequency-based compaction;
- the introduction of Active Child Node Count information to facilitate and accelerate pointers reconstruction;
- the introduction of a variable-length representation of Child Node Mask, with potential lossless compression.

The structure of the paper is as follows:

Section 2 of the paper deals with the linearization of multi-dimensional data using space-filling curves and hierarchical data structures designed for compact representation using lossless compression. Due to the vast number of papers published in this field, this section contains only a selection of works related to the work presented herein.

Section 3 hereof contains an overview of the features, advantages, disadvantages, and explanation of binary-level representations of pointerless SVO and SVDAG. Their favorable properties also represent the benefits of the work described herein.

Section 4 introduces the proposed Pointerless Sparse Voxel Directed Acyclic Graph (PSVDAG) hierarchical data structure, along with a detailed description of its properties, including the concepts of Labels/Callers, variable child node mask length and active child node count, introduced in this work. There is also a detailed description of its binary-level representation.

Section 5 of the paper summarizes the results of tests performed on voxelized 3D scenes with the geometry stored as pointerless SVO, SVDAG, and PSVDAG. Section intends to test the storage efficiency of the data structure proposed in this paper. The examination of the influence of the use of z-order and Hilbert space-filling curves takes part in this section of the paper. It also summarizes sources of compression gains of PSVDAG, compared to SVO and SVDAG.

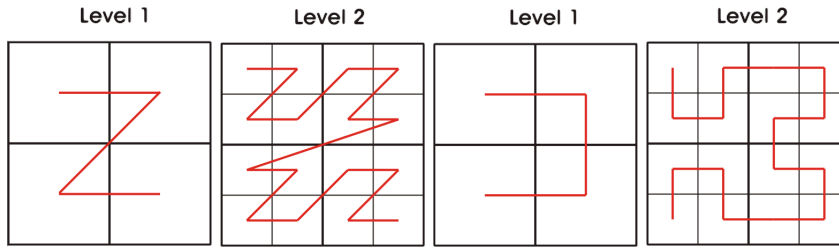
Section 6 summarizes the conclusions drawn from the evaluation of the test results, which details the previous section of this paper.

2 RELATED WORKS

Due to the vast number of papers published in the field of space-filling curves and hierarchical data structures designed to represent multi-dimensional data, this section contains only a selection of publications closely related to the solution proposed and presented herein.

Linearization of multi-dimensional grids of data. Peano and Hilbert presented Space-Filling Curves (SFC) at the end of the 19^{th} century, later followed

by Moore, Lebesgue, Sierpinski, and others [4, 5]. A very popular is the Hilbert Space-Filling Curve (HSFC), introduced by David Hilbert in 1891 [6]. Computer science often uses the Hilbert curve because of its better locality-preserving behavior. The Morton order introduced in 1966 [7], referenced as the Morton sequence, the Morton code, or z-order curve, is often used for the linearized representation of n-dimensional grids of data in computer graphics. As an example of the usage of the z-order curve and Hilbert curve for the filling of two-dimensional space to the different levels, see Figure 1.



a) To the first level b) To the second level c) To the first level d) To the second level

Figure 1. Linearization of two-dimensional space using space-filling z-order curve and alternatively using Hilbert curve

Hierarchical representation of two dimensional data. The utilization of quadtrees to represent images as trees is described in [8, 9, 10, 11]. A discussion on the possibility of using quadtrees for an efficient representation of two-dimensional data using the technique of Common Subtree Merging (CSM) is in [12]. In addition to the empty subsquares uniformly filled with passive pixels, the referenced work also describes the use of subsquares homogeneously filled with active pixels. An appropriate data structure, the Common Subtree Merge Quadtree (CSM-Quadtree), has been created.

The two-dimensional template-based encoding (2DTE) technique, mentioned in [13], was meant for the construction of quadtrees for the use in the cartography field. It represents the lossless compression based on the merging of common subtrees with the use of template mapping and the Morton sequence. This approach extended to three-dimensional voxel data is mentioned in [14].

Hierarchical representation of three-dimensional data. Octrees – a hierarchical data structure – serve to represent three-dimensional data for decades. Works from the 1980s include Srihari [15], Rubin and Whitted [16], Jackins and Tanimoto [17], Meagher [18, 19, 20]. In [21], the author focused on the use of hierarchical data structures such as the octree to speed up the determination of objects intersected by rays emanating from the viewpoint. An algorithm for raytracing octrees consisting of volumetric data is in [22, 23].

Sparse Voxel Octrees (SVO) is a hierarchical data structure designed to represent three-dimensional scenes using octrees. SVO allows for a frugal representation of empty subspaces – the subtrees associated with these empty subspaces are not represented in the data structure.

The octree space decomposition is part of the compression method presented in [24]. It is specialized for point-sampled models explicitly aimed at densely sampled surface geometry.

In 2010, the Efficient Sparse Voxel Octrees (referenced in sources as ESVO), a hierarchical data structure, was introduced in [25]. As an octree hierarchical data structure, it allows carving out empty spaces. It also removes entire subtrees in case there is a possibility to use contour information. It is a representation with a length of 32 b per node, consisting of a 24 b contour pointer and an 8 b contour mask. This modification allows an increase of the geometric resolution, allowing a more compact representation of smooth surfaces, while accelerating ray casting.

In 2013, the Sparse Voxel Directed Acyclic Graph (SVDAG), a hierarchical data structure, was introduced in [26]. SVDAG transforms sparse voxel octrees into oriented acyclic graphs (DAG). This approach uses 8 b masks of child nodes to represent the decomposition of octants to suboctants. The node representation includes a variable number of 32 b pointers pointing to the locations of the active child nodes. This data structure replaces multiple identical subtrees by a single representation of such a subtree. It allows the application of common subtree merging when several child pointers point to the root node of such a subtree. The structure aligns the child node mask to 32 b with unused 24 b (see Figure 2).

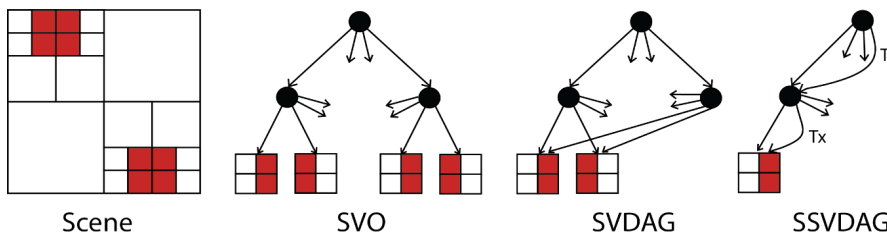


Figure 2. Original scene represented as the 2D grid of pixels (Scene) and as the Sparse Voxel Octree (SVO), Sparse Voxel Directed Acyclic Graph (SVDAG) and Symmetry-aware Sparse Voxel Directed Acyclic Graph (SSVDAG). Tx represents similarity transformation along the axis x, Ty represents similarity transformation along the axis y.

In 2016, an evolution of the SVDAG hierarchical data structure, the Symmetry-aware Sparse Voxel Directed Acyclic Graph (SSVDAG), appeared in [27]. This data structure uses a 16 b child node mask. Each child node describes a 2 b Header Tag (HT) and allows further data compression with a minimal impact

on the decompression overhead. SSVDAG allows merging of common subtrees that are identical when a similarity transformation – a reflection transformation in one or more main axes of a scene – is applied. Another benefit is the use of frequency-based pointer compaction. It orders nodes in each layer of the tree according to their referencing frequency – from the most to the least frequently referenced nodes. The most commonly referenced nodes reside in an area addressed by short, 16 b pointers with a 01 header tag. The less often referenced nodes are in an area addressed by long, 32 b ones with a 10 header tag. Extra-long 33 b pointers have an 11 header tag. A 3 b information – as the part of the vector – is indicating the relevant similarity transformation, as depicted in Figure 3. The last two levels of nodes are forming voxel grids of 4^3 voxels.

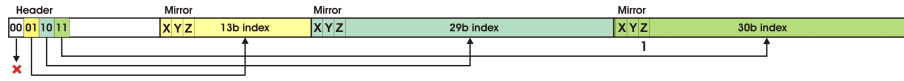


Figure 3. Symmetry-aware Sparse Voxel Directed Acyclic Graph (SSVDAG) binary representation of inner nodes with short, long and extra-long pointers to the child nodes

In 2019, a small modification of the SSVDAG hierarchical data structure, the SSVDAG*, was introduced in [28]. As part of this modification, pointers with the header tag 11, which allowed 33-bit addresses and an indication of reflective transformations – were replaced with 16-bit ones with an identical header tag without transformation indication. Compared to short 16-bit ones of the SSVDAG, these allow an 8-fold increase in address space, provided there is no need for reflective transformations. Thus, some long, 32-bit pointers become 16-bit, with the advantage that these point to more frequently referenced nodes. Compared to SSVDAG, this allows further compression. On the other hand, a reduction of the total addressable space occurs.

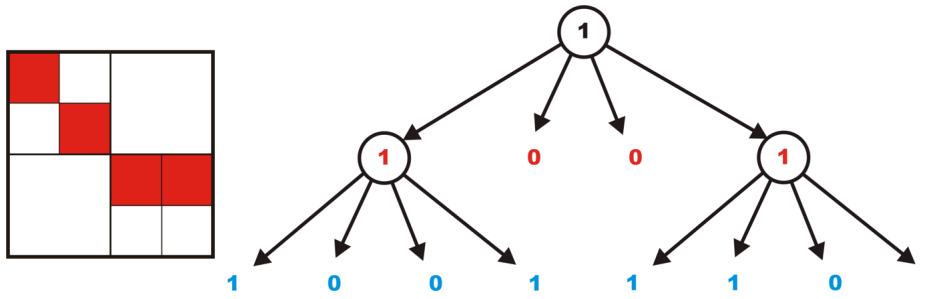
Out-of-core construction algorithms. Out-of-core construction of Sparse Voxel Octrees from triangle meshes was introduced in [29] in 2013. The algorithm consists of two basic steps. The first is a voxelization process. It transforms the triangles representing the scene to the intermediate product – a high-resolution three-dimensional voxel grid. In the second step, the transformation of this intermediate product into Sparse Voxel Octree (SVO) occurs. The algorithm allows the size of the binary representation of the input mesh of triangles, the output SVO, and the intermediate product – a three-dimensional voxel grid generated at a high resolution and represented by a Morton order – to exceed the available computer operating memory by far. Compared to the in-core algorithm, it uses only a fraction of the memory while maintaining the use of comparable time.

3 HIERARCHICAL DATA STRUCTURES

This section presents a brief overview of pointerless Sparse Voxel Octrees (SVO) and Sparse Voxel Directed Acyclic Graphs (SVDAG). It provides information about their basic properties, advantages, disadvantages, and binary-level encoding.

3.1 Pointerless Sparse Voxel Octrees Overview

Sparse Voxel Octrees (SVO) are hierarchical data structures enabling the representation of a voxelized three-dimensional scene with a frugal way of encoding empty subspaces. The SVO root node represents the entire three-dimensional scene. SVO divides this space into eight octants and contains one bit for each of them. This bit is set to 0 if the octant does not comprise any active voxels (i.e., the octant is empty); thus, it will not decompose. This bit is set to 1 if the particular octant contains at least one active voxel. If this active octant consists of more than a single voxel, decomposition to suboctants follows recursively. The pointerless version of SVO does not contain any pointers to child nodes.



a) A two-dimensional pixel grid

b) An SVO in tree form

1(1(1001)001(1100))

c) A binary representation of SVO using parentheses to illustrate the recursive decomposition of active sub-quadrants

11100100111 00

d) The final binary SVO representation

Figure 4. An illustration of the SVO hierarchical data structure to simplify the example, used to encode a two-dimensional pixel grid using a z-order curve for linearization

Figure 4 a) depicts a two-dimensional grid having 4×4 pixels, with the active pixels indicated in red, and the inactive pixels in white. SVO hierarchical data structure, in the form of the tree, is shown in Figure 4 b). There are active quadrants 0 and 3, subsequently decomposed. The binary representation of the pointerless SVO structure – with parentheses added to illustrate the decomposition points – is in Figure 4 c). The final binary encoding of the pointerless SVO data structure for

this pixel grid, is stated in Figure 4 d). In the hierarchical data structure, each decomposition of the tree nodes generated four bits of information concerning the potential child nodes. With three-dimensional voxel grids, there are 8 bits of eight possible child nodes for each octant decomposition of the respective octant. Those 8 bits are forming the Child Node Mask (CHNM).

The advantage of the hierarchical data structure constructed as stated above is its compact binary representation, since each node decomposition generated only eight bits. Pointerless data structure has its advantage in the absence of pointers to child nodes that represent vast amounts of data. In the most unfavorable case, if there are all eight pointers to child nodes, using 32 bits per pointer, an additional 256 b would have to be allocated. The pointerless data structure constructed, as stated above, is suitable for streaming or archiving purposes. The downside is in its cumbersome and time-consuming traversing at the time of scene visualization. Another disadvantage of such a data structure is that it does not allow the compact representation using CSM.

3.2 Sparse Voxel Directed Acyclic Graph Overview

The Sparse Voxel Directed Acyclic Graph (SVDAG) is a hierarchical data structure in the form of a directed acyclic graph. It consists of nodes containing child nodes masks of these nodes and the corresponding pointers to child nodes. The node mask is composed of 8 bits, assigned to the respective suboctants. A bit set to 0 represents an empty suboctant, containing no active voxel. In this case, in the data structure, the corresponding child node – including the child node pointer – is not present. A bit set to 1 represents an active suboctant, containing at least one active voxel. For such a suboctant, a child node is created in the data structure, referenced by the corresponding node pointer. The concatenation of 24 unused bits aligns the mask to 32 bits.

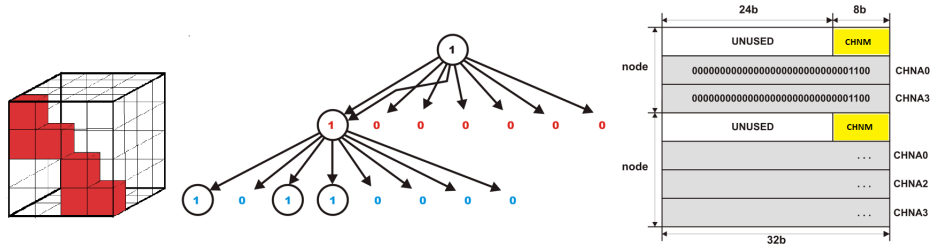
A block of pointers to the existing child nodes follows the node mask. Their count is from the range $\langle 1; 8 \rangle$. Each one has 32 bits and points to a memory location where a binary representation of the corresponding child node is stored. All node parts and thus, each node and the whole data structure are aligned to 32 bits, as shown in Figure 5 c). Multiple pointers of different but also the same node may point to the same memory location. Therefore multiple identical subDAGs may be represented by fully encoding only a single copy of subDAG, with several references to it by child node pointers. SVDAG thus enables CSM.

The length of the SVDAG node binary-level representation l having n pointers to child nodes can be calculated as

$$l = (n + 1) \times 32 \text{ [b]}. \tag{1}$$

Compared to pointerless SVOs, the advantage of the SVDAG data structure lies in the introduction of pointers to child nodes, which allows fast graph traversal when processing and visualizing the scene. Another advantage is the possibility of merging

common subtrees, through multiple references to a particular node by several child node pointers. It allows significant compaction of the tree's binary representation. It is a compression method that does not reduce the speed of graph traversal, compared to the version without common subtrees merging implementation.



a) A grid of voxels b) An SVDAG as a graph c) The final binary representation of SVDAG

Figure 5. An example of the binary representation of three-dimensional space using a Sparse Voxel Directed Acyclic Graph (SVDAG) and an z-order curve linearization

The disadvantage of the SVDAG data structure, in comparison to SVO, is that it consumes large amounts of secondary storage or memory. Due to the unused 24 bits used for aligning the child node mask (CHNM) to 32 bits, compared to the 8 bits of SVO. Another disadvantage may be the introduction of 32 b child node pointers whose binary representation may be unnecessarily long – these are not present in pointerless SVOs. However, this disadvantage compensates multiple referencing to common subtrees. It can offset these disadvantages to the extent that for a particular three-dimensional grid, the binary representation of an SVDAG may be more compact than that of an SVO.

Figure 5 a) shows an example of a three-dimensional voxel grid, while Figure 5 b) shows a Sparse Voxel Directed Acyclic Graph. The binary representation of SVDAG shown in Figure 5 c), where the first node is decomposed and mask for its child nodes is "00001001", with two non-empty octants, represented by bits 1. For each non-empty node, there is a 32-bit address (CHNA0 and CHNA3) pointing to the memory location where the child node binary representation is stored. Since those nodes are identical, both pointers are referencing the same address in memory. Another node, referenced two times, has the mask "00001101", i.e., it has three active octants. That is why three addresses (CHNA0, CHNA2, and CHNA3) are present for the child nodes (without actual addresses in this example).

4 POINTERLESS SPARSE VOXEL DIRECTED ACYCLIC GRAPH

The hierarchical data structure, the Pointerless Sparse Voxel Directed Acyclic Graph (PSVDAG), proposed in this paper, is a directed acyclic graph. It represents a three-dimensional grid of voxels R, organized as $N \times N \times N$ voxels. $R = N^3$, where $N = 2^n$,

$n \in \mathbb{N}$. Each voxel is represented by 1 b, when active voxel is encoded by the value '1' and passive voxel by the value '0'. For storing the grid R in uncompressed form, there is a need for the space, which size S is:

$$S = 2^{3n} \text{ [b]}. \tag{2}$$

When represented by PSVDAG hierarchical data structure, the grid R mentioned above is encoded into nodes stored in n levels. Each level has a level mark l with the value from the range of $\langle 0; n-1 \rangle$. The root node is in the level, where $l = 0$. Nodes stored in levels which level mark $l \leq n-2$ have the same structure, described in the Section 4.1 – these are the Inner nodes. Leaf nodes, i.e., nodes stored in the level $l = n-1$, have a different structure, described in Section 4.2.

Formal description of the PSVDAG using Backus-Naur Form (BNF) is as follows:

PSVDAG ::= ⟨NODE⟩
NODE ::= ⟨INODE⟩ | ⟨LNODE⟩
INODE ::= ⟨ACHNC⟩⟨CHNM⟩
ACHNC ::= (3)⟨BIT⟩
CHNM ::= (1) ∗ (8)⟨HT⟩
HT ::= "00" | "01"⟨LAB⟩⟨NODE⟩ | "10"⟨CAL⟩ | "11"⟨NODE⟩
LAB ::= ⟨SIZ⟩⟨VAL⟩
CAL ::= ⟨SIZ⟩⟨VAL⟩
SIZ ::= (5)⟨BIT⟩
VAL ::= (1) ∗ (32)⟨BIT⟩
LNODE ::= (8)⟨BIT⟩
BIT ::= "0" | "1"

where following is applied:

- ⟨SYM⟩ – mandatory symbol SYM,
- "sym" – terminal symbol *sym*,
- $(n)\langle \text{SYM} \rangle$ – symbol SYM concatenated n times,
- $(n) * (m)\langle \text{SYM} \rangle$ – symbol SYM concatenated from n to m times,
- | – alternative,
- *juxtaposition* – the concatenation.

An example in Figure 6 is, for the sake of simplicity, based on the two-dimensional grid of 4×4 pixels. Related quadtree is constructed along with the PSVDAG, using two different space-filling curves (z-order and Hilbert) for linearization.

Formal description of the PSVDAG for two-dimensional grid includes the following changes:

ACHNC ::= (2)⟨BIT⟩
CHNM ::= (1) ∗ (4)⟨HT⟩
LNODE ::= (4)⟨BIT⟩

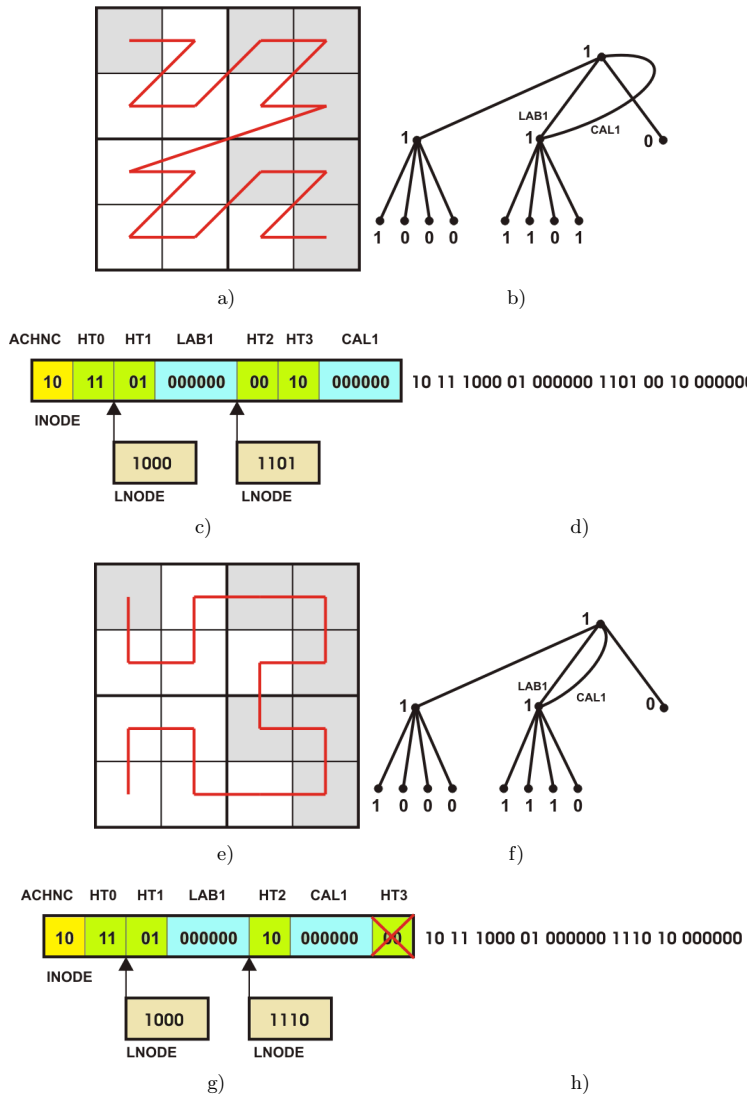


Figure 6. An example of a) 2D grid of pixels and z-order space-filling curve, b) related SVDAG, c) PSVDAG, and, d) final binary representation of PSVDAG. An example of e) 2D grid of pixels and Hilbert space-filling curve, f) related SVDAG, g) PSVDAG, and, h) final binary representation of PSVDAG. ACHNC – Active Child Node Count, HTi – Header Tag, LAB1 – Label, CAL1 – Caller, INODE – Inner Node, LNODE – Leaf Node. SVDAG includes denotations LAB1 and CAL1 to describe the relationship to PSVDAG construction better.

4.1 Nodes

Node of the data structure represents grid $R' : R' \subseteq R$ of voxels, where $R' = N'^3$, $N' = 2^m$, $m \in \mathbb{N} \wedge m \leq n$. There is possibility to find $t = N'/2$ and R' can be divided into 8 octants, forming the set OCT:

$$\text{OCT} = \{oct_0, oct_1, oct_2, \ldots, oct_5, oct_6, oct_7\} \tag{3}$$

where

$$\begin{aligned}
oct_0 &= \langle 0; t-1\rangle \times \langle 0; t-1\rangle \times \langle 0; t-1\rangle,\\
oct_1 &= \langle t; N'-1\rangle \times \langle 0; t-1\rangle \times \langle 0; t-1\rangle,\\
oct_2 &= \langle 0; t-1\rangle \times \langle t; N'-1\rangle \times \langle 0; t-1\rangle,\\
oct_3 &= \langle t; N'-1\rangle \times \langle t; N'-1\rangle \times \langle 0; t-1\rangle,\\
oct_4 &= \langle 0; t-1\rangle \times \langle 0; t-1\rangle \times \langle t; N'-1\rangle,\\
oct_5 &= \langle t; N'-1\rangle \times \langle 0; t-1\rangle \times \langle t; N'-1\rangle,\\
oct_6 &= \langle 0; t-1\rangle \times \langle t; N'-1\rangle \times \langle t; N'-1\rangle,\\
oct_7 &= \langle t; N'-1\rangle \times \langle t; N'-1\rangle \times \langle t; N'-1\rangle.
\end{aligned}$$

Octants $oct_i \in \text{OCT} : i \in \langle 0; 7\rangle$ are ordered in the OCT set according to selected space-filling curve.

When $R' = R$; $m = n$, the root node of the PSVDAG is constructed.

When $m > 1$, the node is constructed as the Inner node INODE.

When $m = 1$, the node is constructed as the Leaf node LNODE.

4.2 Inner Nodes

Inner node consists of Active Child Node Count (ACHNC) followed by the Child Node Mask (CHNM):

INODE ::= ⟨ACHNC⟩⟨CHNM⟩

4.2.1 Active Child Node Count

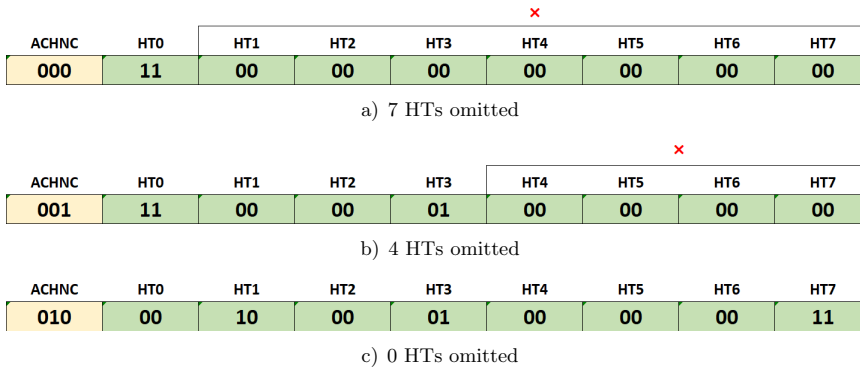
Inner node includes Active Child Node Count (ACHNC) information:

ACHNC ::= (3)⟨*BIT*⟩

It is a 3-bit vector forming unsigned integer number that is referring to the number of active child nodes of a given node. Active child nodes are those having identification '01', '10' or '11' in their HTs, that are part of Inner nodes Child Node Mask (CHNM). An Inner Node cannot have 0 active child nodes, therefore count

range is $\langle 1; 8\rangle$. Decrementing the number of active child nodes by the value 1 allows us to achieve the ACHNC range of $\langle 0; 7\rangle$ to represent it on 3 bits. Thus, if the count of active child nodes is c, the binary representation of ACHNC is set to an unsigned integer of the value $c - 1$.

When sequentially browsing the HTs of the particular nodes CHNM from the HT at position 0 (HT0) to the HT at position 7 (HT7), ACHNC allows us to identify last active child node HTx in their sequence. All subsequent HT identifications, i.e., HT$z : z > x$ must be set to '00'. It is possible to derive this information from the ACHNC and the sequence of previous HTs. That is why there is no need to represent HT$z : z > x$ in the CHNM, and it is possible to omit them. It allows to compress the CHNM (See Figure 7).



a) 7 HTs omitted

b) 4 HTs omitted

c) 0 HTs omitted

Figure 7. Examples of ACHNC and concatenated CHNM containing HTs (for simplicity represented only by their identifications) indicate

There is an unfavorable side effect in the time of reconstructing the pointers or converting PSVDAG to SVDAG. The individual HTs identifications in the CHNM of the PSVDAG node, in general, do not form a continuous vector of bits in the linearized binary representation. Therefore, the ACHNC has been added to the data structure to facilitate and accelerate pointers reconstruction. When the processing of a particular PSVDAG node begins, the ACHNC can be used to determine the number of active child nodes of the SVDAG node. Subsequently, also to determine the space needed to store related pointers. Another advantage is that there is a possibility to determine when the last active child node HT has been processed and immediately finish the node processing.

4.2.2 Child Node Mask

Child node mask consists of Header Tags (HT), HT$_i : i \in \langle 0; 7\rangle$, which are assigned to octants oct$_i \in$ OCT $: i \in \langle 0; 7\rangle$ in ascending order. Encoding of the CHNM and HT is as follows:

CHNM ::= $(1) * (8)\langle \text{HT} \rangle$
HT ::= "00" | "01"$\langle \text{LAB} \rangle\langle \text{NODE} \rangle$ | "10"$\langle \text{CAL} \rangle\rangle$ | "11"$\langle \text{NODE} \rangle$

Passive child node HT. Octant $oct_i \in \text{OCT} : i \in \langle 0; 7 \rangle$ is passive when it does not contain any active voxels. The HT is, in that case, represented only by identification symbol "00". No further information follows (Passive child node header tags can be omitted, as mentioned above in the Section 4.2.1).

Active child node HT. Octant $oct_i \in \text{OCT} : i \in \langle 0; 7 \rangle$ is active when its grid contains at least one active voxel.

If octant $oct_i \in \text{OCT} : i \in \langle 0; 7 \rangle$ is not unique (in related SVDAG its node is referenced by more than one pointer), and it will serve as the template, the HT is represented by symbol "01". Label LAB concatenates along with the NODE, constructed for the grid of this octant.

If octant $oct_i \in \text{OCT} : i \in \langle 0; 7 \rangle$ is not unique (in related SVDAG, there is more than one pointer to this node), and it will not serve as the template, the HT is represented by symbol "10". Caller CAL concatenates.

If octant $oct_i \in \text{OCT} : i \in \langle 0; 7 \rangle$ is unique (in related SVDAG, there is only one pointer to this node), the HT is represented by symbol "11". Binary representation of the NODE constructed for the grid of this octant concatenates.

When octant $oct_i \in \text{OCT} : i \in \langle 0; 7 \rangle$ is not unique (it is referenced in related SVDAG x times), its binary representation in PSVDAG is fully developed only once, when it is referenced by the assigned label LAB_d to which the octants NODE is concatenated. The octant is further referenced $x - 1$ times using caller CAL_d. Binary representation of $\text{LAB}_d = \text{CAL}_d$.

That is how the set P of x pointers to the particular common subDAG of SVDAG is replaced by one label LAB_d followed by the binary representation of this subDAG and $x - 1$ times by the caller CAL_d. The first used pointer from the set P, when traversing SVDAG in Depth-First order, is replaced by a label followed by the binary representation of subDAG. Caller will replace the other pointers from the set P. It ensures that when traversing PSVDAG in the Depth-First order, the label LAB_d will be used before any of occurrences of the caller CAL_d – see Figure 8.

4.2.3 Labels and Callers

Labels in the PSVDAG data structure identify the corresponding octants. Binary encoding of the label is respecting the rules:

LAB ::= $\langle \text{SIZ} \rangle\langle \text{VAL} \rangle$
SIZ ::= $(5)\langle \text{BIT} \rangle$
VAL ::= $(1) * (32)\langle \text{BIT} \rangle$
BIT ::= "0" | "1"

SIZ is a 5-bit vector representing an unsigned integer from the range of $\langle 0 : 31 \rangle$. Incremented by value 1, it is representing the number of bits of the concatenated

VAL. Provided the SIZ is 0, the length of the binary representation of the VAL is 1 b.

VAL is the (SIZ + 1) – bit vector, that is representing unsigned integer value from the range of $\langle 0; 2^{SIZ+1} - 1\rangle$.

When set of labels is generated for the PSVDAG data structure, (for each level l separately), the SIZ = 0 and VAL = 0, for the label $\text{LAB}_i : i = 0$. For each next label the VAL is incremented by 1, until the value of the $\text{VAL} = 2^{\text{SIZ}+1} - 1$. For next label VAL = 0 and SIZ is incremented by 1.

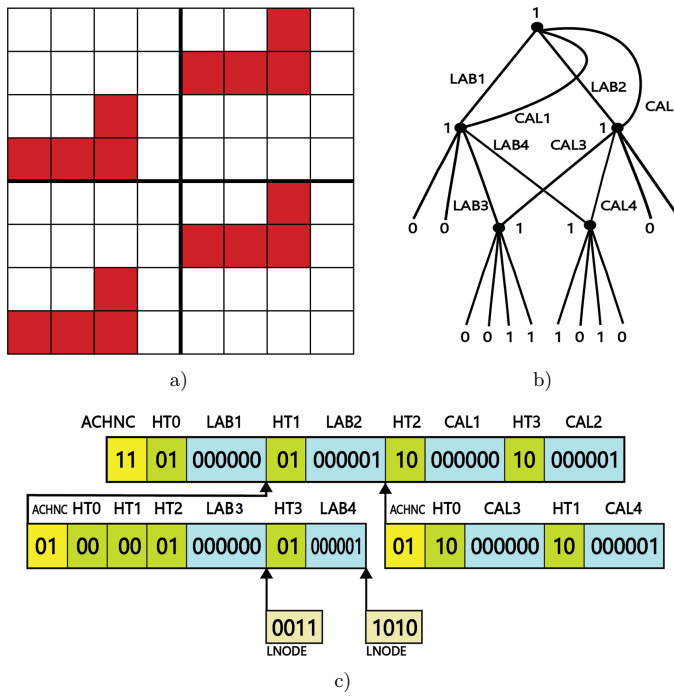


Figure 8. Example of a) two-dimensional grid of 8×8 pixels, b) SVDAG with denotations of LABs and CALs for better description of c) PSVDAG

It is possible to have two different labels with the same SIZ and VAL values, i.e., LAB_a and LAB_b, where $\text{SIZ}_a = \text{SIZ}_b \wedge \text{VAL}_a = \text{VAL}_b$, if those labels are located in different levels l of the data structure. Provided there is l, SIZ, and VAL concatenated, each label is unique in the data structure. There is no need to represent level l value in the PSVDAG because it is possible to derive it from the data structure in the traversing time.

Callers are constructed in the same way as the labels, using the rules:

CAL ::= ⟨SIZ⟩⟨VAL⟩
SIZ ::= (5)⟨BIT⟩

VAL ::= (1) ∗ (32)⟨BIT⟩
BIT ::= "0" | "1"

Caller CAL_C with the level value l_C, SIZ = siz_C, VAL = val_C is referencing the same node as the LAB_L with the level value l_L, SIZ = siz_L, VAL = val_L when $l_C = l_L \wedge siz_C = siz_L \wedge val_C = val_L$.

4.2.4 Frequency-Based Compaction

Particular nodes in PSVDAG data structure at level l are referenced from the level $l-1$ different times, using labels and callers (references). Each label/caller can have variable length of the binary representation. Different permutations of labels/callers assignment to particular nodes generate different overall sum of labels/callers lengths. Therefore frequency-based compaction is applied to find out permutation that has the smallest overall sum of reference lengths.

It is possible to form the NOD set consisting of p nodes from level l of PSVDAG that are each referenced more than once. Members of the set are in descending order according to the frequency of their references.

$$\mathrm{NOD} = \{nod_0, nod_1, nod_2, \ldots, nod_{p-3}, nod_{p-2}, nod_{p-1}\}. \tag{4}$$

Generated labels – unique one for each $nod_i \in \mathrm{NOD} : i \in \langle 0; p-1\rangle$ – form the set LAB that consists of p labels, that have different lengths of their encoding. The LAB set ordering is ascending order according to the length of the binary representation of particular labels.

$$\mathrm{LAB} = \{lab_0, lab_1, lab_2, \ldots, lab_{p-3}, lab_{p-2}, lab_{p-1}\}. \tag{5}$$

Using frequency-based compaction, the node $nod_i \in \mathrm{NOD} : i \in \langle 0; p-1\rangle$ is assigned by the label $lab_j \in \mathrm{LAB} : j \in \langle 0; p-1\rangle$, $i = j$.

Each node assigned by the label $lab_i \in \mathrm{LAB} : i \in \langle 0; p-1\rangle$ with the length of binary representation len is referenced by the caller (used one or more times) that has the same length of the binary representation len.

An example of different permutations of label assignments, including one based on frequency-based compaction, can be seen in Figure 9.

4.3 Leaf Nodes

By the Leaf node, the octant, which grid $R' = 2^3$ (containing 8 voxels), is encoded. Leaf node consists of 8 bit vector where only information about voxels is stored. Passive voxels are represented using bit 0, active voxels are represented using bit 1. Linearization of the grid R' is performed according to chosed space-filling curve, as it is depicted on Figure 10.

Node id	1	2	3	4	5	
Referencing frequency	2	1	3	8	1	
Reference [b]	6	6	7	7	7	
Sum [b]	12	6	21	56	7	∑ = 102

Node id	4	3	1	2	5	
Referencing frequency	8	3	2	1	1	
Reference [b]	6	6	7	7	7	
Sum [b]	48	18	14	7	7	∑ = 94

Figure 9. An example of two different permutations of label assignments to the respective nodes of a particular PSVDAG level (for clarity, Node id is added in this example). The top part of the figure represents arbitrary node order, and the overall sum of labels/callers length is 102 b, the bottom part represents node order achieved using frequency-based compaction, and in this case, the overall sum of labels/callers length is 94 b.

5 RESULTS AND DISCUSSION

Various 3D scenes, originally stored in Wavefront OBJ geometry definition file format (examples of those scenes with their visualizations are in Figure 11), were used for test purposes. Voxelization was performed to the different resolutions ranging from 128^3 to the $4\,096^3$ ($4K^3$). Voxelized scenes were encoded to SVO, SVDAG, and PSVDAG hierarchical data structures. Evaluation of results obtained by tests

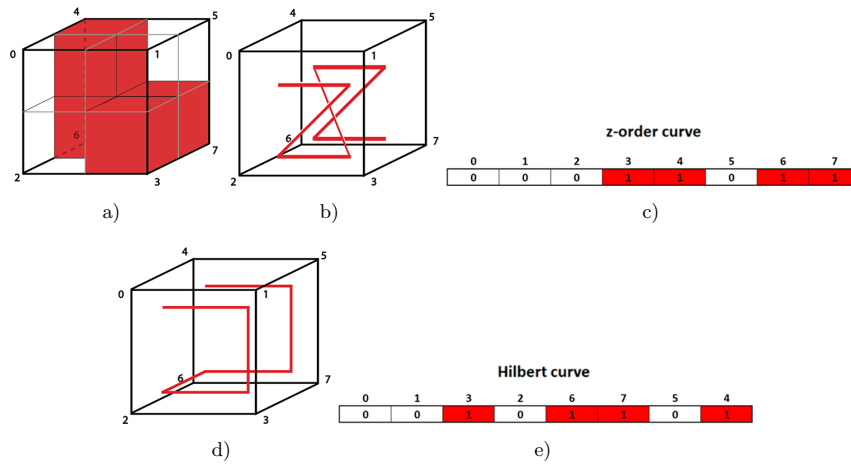
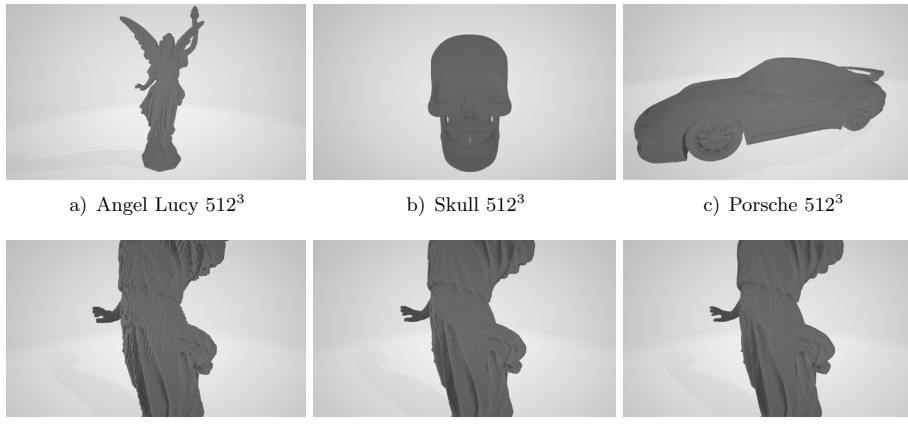


Figure 10. a) Leaf node consisting of 4 active voxels (red), b) z-order curve, c) voxels in binary representation of the node arranged according to z-order curve, d) Hilbert curve, e) voxels in binary representation of the node arranged according to Hilbert curve

followed. Tests were performed on the computer using four core CPU Intel® Core i5-8265U at 1.6 GHz, 8 GB operating memory, NVIDIA GeForce GTX 1050 3 GB graphics card, and 256 GB SSD as the secondary storage.



a) Angel Lucy 512^3 b) Skull 512^3 c) Porsche 512^3

d) Detail of the Angel Lucy 256^3 e) Detail of the Angel Lucy 512^3 f) Detail of the Angel Lucy $1K^3$

Figure 11. Visualization of examples of voxelized scenes used for testing purposes

The test results show that the representation of the scene geometry using the PSVDAG data structure is significantly smaller than in SVDAG. It applies to all tested three-dimensional voxelized data sets and the z-order and Hilbert curve linearizations. In the most optimistic case, at a 128^3 resolution, a compression ratio of 3.77 was reached. The binary representation of this scene in PSVDAG thus represents only 26.5 % of the size of its binary encoding in SVDAG. Generally speaking, an increase in the resolution leads to a gradual decrease in the compression ratio. It reached the value 2.80 at a resolution of $4\,096^3$, when the binary representation in PSVDAG occupies only 35.7 % of the corresponding encoding size in SVDAG.

The size of the binary representation in PSVDAG is, with increasing resolution, relatively decreasing, compared to SVO. In most cases, PSVDAG outperformed SVO for both, z-order curve as well as Hilbert curve linearizations. In these tests, PSVDAG to the corresponding SVO size ratio, is ranging from 131.1 % to 54.5 %. The compression ratio is also shown separately in Figures 12 and 13 for z-order curve linearization.

A comparison in terms of absolute sizes of binary representations is available in Table 1 for z-order curve linearization and in Table 2 for the Hilbert curve. The PSVDAG-SVDAG and PSVDAG-SVO compression ratios are available in Tables 3 and 5. This also shows the relative space requirements of the PSVDAG-encoded scene, compared to the SVDAG and SVO versions. The PSVDAG-SVDAG compression ratio, when the z-order curve is applied, is comparable to the version using the Hilbert curve, for all models and resolutions. There are only small fluctuations in the range of $\langle -0.687\,\%; 1.657\,\% \rangle$. Size comparison showed a slightly higher success

Size [KB]	Resolution					
	128^3	256^3	512^3	$1K^3$	$2K^3$	$4K^3$
Angel Lucy						
SVO	6.67	28.62	117.99	475.98	1 895.02	7 447.62
SVDAG	31.88	127.01	462.85	1 523.74	4 682.54	14 928.34
PSVDAG	8.62	35.10	132.37	462.04	1 530.86	5 208.41
Skull						
SVO	23.24	95.61	387.45	1 551.55	6 129.76	23 667.56
SVDAG	100.39	356.75	1 182.45	3 728.80	12 275.23	43 086.88
PSVDAG	28.11	104.31	366.48	1 239.06	4 320.48	15 279.20
Porsche						
SVO	14.59	67.52	295.10	1 241.50	5 087.54	20 262.88
SVDAG	56.78	222.41	788.76	2 629.21	8 918.93	32 127.96
PSVDAG	15.08	61.28	227.13	800.89	2 894.32	11 048.85

Table 1. Comparison of SVO, SVDAG and PSVDAG hierarchical data structures using different scenes and resolutions in terms of data structure size (their binary representation in KB) – linearized using z-order curve

in common subtree merges when z-order linearization is applied. Subsequently, the memory consumption and the time for the compression of SVDAG data structure were also smaller, as shown in Tables 7 and 8.

Adding Active Child Node Count information allows potential compression of the Child Node Mask. Hence, the authors monitored whether the 16 b of identifications of Child Node Mask was compressed or inflated in the tested scenes. The

Size [KB]	Resolution					
	128^3	256^3	512^3	$1K^3$	$2K^3$	$4K^3$
Angel Lucy						
SVO	6.67	28.62	117.99	475.98	1 895.02	7 447.62
SVDAG	32.14	128.25	477.07	1 596.11	4 922.56	15 519.44
PSVDAG	8.74	36.00	137.74	486.17	1 613.09	5 434.41
Skull						
SVO	23.24	95.61	387.45	1 551.55	6 129.76	23 667.56
SVDAG	101.55	362.38	1 197.48	3 780.68	12 447.23	43 745.21
PSVDAG	28.44	105.87	371.02	1 254.54	4 370.65	15 621.87
Porsche						
SVO	14.59	67.52	295.10	1 241.50	5 087.54	20 262.88
SVDAG	59.99	234.71	836.33	2 802.36	9 480.74	34 082.36
PSVDAG	15.92	64.75	240.71	852.27	3 064.59	11 637.51

Table 2. Comparison of SVO, SVDAG and PSVDAG hierarchical data structures using different scenes and resolutions in terms of data structure size (their binary representation in KB) – linearized using Hilbert curve

Parameter	**Resolution**					
	128^3	256^3	512^3	$1K^3$	$2K^3$	$4K^3$
Angel Lucy						
PSVDAG in comparison to SVO						
Compression	0.77	0.82	0.89	1.03	1.24	1.43
Percents	129.3 %	122.6 %	112.2 %	97.1 %	80.8 %	69.9 %
PSVDAG in comparison to SVDAG						
Compression	3.70	3.62	3.50	3.30	3.06	2.87
Percents	27.0 %	27.6 %	28.6 %	30.3 %	32.7 %	34.9 %
Skull						
PSVDAG in comparison to SVO						
Compression	0.83	0.92	1.06	1.25	1.42	1.55
Percents	120.9 %	109.1 %	94.6 %	79.9 %	70.5 %	64.6 %
PSVDAG in comparison to SVDAG						
Compression	3.57	3.42	3.23	3.01	2.84	2.82
Percents	28.0 %	29.2 %	31.0 %	33.2 %	35.2 %	35.5 %
Porsche						
PSVDAG in comparison to SVO						
Compression	0.97	1.10	1.30	1.55	1.76	1.83
Percents	103.4 %	90.7 %	77.0 %	64.5 %	56.9 %	54.5 %
PSVDAG in comparison to SVDAG						
Compression	3.76	3.63	3.47	3.28	3.08	2.91
Percents	26.6 %	27.6 %	28.8 %	30.5 %	32.5 %	34.4 %

Table 3. Comparison of compression ratios and size percentages of binary representations of PSVDAG to SVO and PSVDAG to SVDAG hierarchical data structures, using different scenes and resolutions – linearized using z-order curve

tests showed that the size of the child node masks binary representation increased in most cases – its average length is from the range of $\langle 15.82; 17.34\rangle$ b. Thus, in some models, the average size of the child node mask was reduced up to 98.88 % in comparison to 16 b size. In some models, inflation was evident (in this case, the most pessimistic case showed a 108.38 % compared to the 16 b size.) However, even in the most pessimistic case, the combined size of ACHNC and CHNM was smaller than the CHNM of the SVDAG data structure, accompanied by the reserved 24 b. In this case, the worst compression ratio reached for this part of the node was 1.85 (54.19 % of the original size). The relevant data is available in Tables 4, 6, and Figure 14.

The average length of the binary representation of labels/callers gradually increases with an increased resolution of the voxelized 3D scene because of the rising number of labels. For example, at the lowest resolution of 128^3, the Angel Lucy model for z-order averaged 8.75 b per label/caller, and 8.90 b in case of Hilbert curve. It is a 3.66 (27.34 %), resp. 3.60 (27.81 %) compression ratio, compared to the 32 b SVDAG pointers. The maximum length of the label/caller binary repre-

Size [b]	Resolution					
	128^3	256^3	512^3	$1K^3$	$2K^3$	$4K^3$
Angel Lucy						
Child Node Mask	15.89	16.73	17.11	17.28	17.33	17.30
Labels and Callers	8.75	8.90	9.21	9.86	10.87	11.93
Skull						
Child Node Mask	16.82	17.12	17.29	17.35	17.33	17.34
Labels and Callers	9.01	9.43	10.11	11.06	12.03	12.49
Porsche						
Child Node Mask	16.89	17.23	17.43	17.44	17.39	17.32
Labels and Callers	8.24	8.66	9.17	9.91	10.83	11.79

Table 4. Selected parameters of PSVDAG hierarchical data structure according to the different scenes and their resolutions – linearized using z-order curve

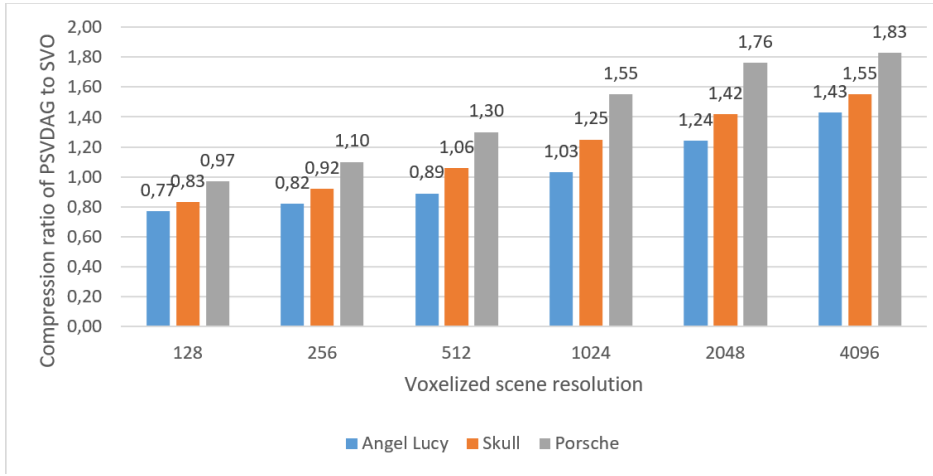


Figure 12. The PSVDAG-SVO compression ratio for each scene and the resolutions used to voxelize the respective scenes, when z-order applied

sentation for this model reached a maximum at the resolution of $4\,096^3$ when the average label/caller size was 11.93 b resp. 11.96 b. It represents a 2.68-fold compression compared to the 32 b SVDAG pointers and reduction to 37.28 % resp. 37.38 % in terms of size. The numerical values of this parameter for the respective models and resolutions are in Tables 4, 6, and Figure 15.

5.1 Sources of Compression Gains in PSVDAG

The PSVDAG data structure provides several data compression sources, when compared to the SVDAG:

Parameter	**Resolution**					
	128^3	256^3	512^3	$1K^3$	$2K^3$	$4K^3$
Angel Lucy						
PSVDAG in comparison to SVO						
Compression	0.76	0.80	0.86	0.98	1.17	1.37
Percents	131.1 %	125.8 %	116.7 %	102.1 %	85.1 %	73.0 %
PSVDAG in comparison to SVDAG						
Compression	3.68	3.56	3.46	3.28	3.05	2.86
Percents	27.2 %	28.1 %	28.9 %	30.5 %	32.8 %	35.0 %
Skull						
PSVDAG in comparison to SVO						
Compression	0.82	0.90	1.04	1.24	1.40	1.52
Percents	122.4 %	110.7 %	95.8 %	80.9 %	71.3 %	66.0 %
PSVDAG in comparison to SVDAG						
Compression	3.57	3.42	3.23	3.01	2.85	2.80
Percents	28.0 %	29.2 %	31.0 %	33.2 %	35.1 %	35.7 %
Porsche						
PSVDAG in comparison to SVO						
Compression	0.92	1.04	1.23	1.46	1.66	1.74
Percents	109.1 %	95.9 %	81.6 %	68.6 %	60.2 %	57.4 %
PSVDAG in comparison to SVDAG						
Compression	3.77	3.62	3.47	3.29	3.09	2.93
Percents	26.5 %	27.6 %	28.8 %	30.4 %	32.3 %	34.1 %

Table 5. Comparison of compression ratios and size percentages of binary representations of PSVDAG to SVO and PSVDAG to SVO hierarchical data structures, using different scenes and resolutions – linearized using Hilbert curve

- The child nodes mask of the SVDAG data structure has a constant size of 8 b (i.e., 1 b per octant), with another 24 b unused for the sake of aligning to 32 b. That results in a total of 32 b, i.e., 4 b per octant. PSVDAG uses a 3-bit Active Child Node Count. The Child Node Mask identifications itself have a binary representation with a total length ranging from 2 b to 16 b, with 2 b per potential child node. The sum of ACHNC and CHNM ranges from 5 b to 19 b, which is significantly less than that of SVDAG. 8 octants use space ranging from 0.625 b per child node to 2.375 b per child node. The compression ratio (in comparison to SVDAG) is from the range of 1.68 to 6.4. The reduction of this part of the data structure is to a level of 15.63 % to 59.38 %.
- The absence of child node pointers in PSVDAG is a significant source of its reduction, considering the binary representation (compared to the SVDAG data structure). If the subgraph is unique, there is no pointer at its root node in the data structure. Thus the 32 b pointer of SVDAG becomes 0 b in PSVDAG.
- If the particular subgraph is not unique in the data structure, the 32 b pointer of the SVDAG, pointing at the root node of the subgraph, is replaced by a la-

Size [b]	Resolution					
	128^3	256^3	512^3	$1K^3$	$2K^3$	$4K^3$
Angel Lucy						
Child Node Mask	15.82	16.69	17.08	17.28	17.34	17.32
Labels and Callers	8.90	9.12	9.37	9.95	10.90	11.96
Skull						
Child Node Mask	16.75	17.07	17.26	17.30	17.27	17.27
Labels and Callers	9.04	9.44	10.11	11.05	12.00	12.41
Porsche						
Child Node Mask	16.80	17.14	17.31	17.35	17.30	17.25
Labels and Callers	8.30	8.71	9.21	9.92	10.80	11.70

Table 6. Selected parameters of PSVDAG hierarchical data structure according to the different scenes and their resolutions – linearized using Hilbert curve

Size [KB]	Resolution					
	128^3	256^3	512^3	$1K^3$	$2K^3$	$4K^3$
Angel Lucy						
z-order curve	39.48	156.13	544.49	1 635.08	4 225.91	10 801.93
Hilbert curve	39.74	157.82	566.83	1 752.33	4 617.10	11 745.66
Skull						
z-order curve	120.63	406.15	1 224.49	3 244.12	8 718.88	33 376.02
Hilbert curve	122.30	414.52	1 245.42	3 320.14	8 967.54	34 912.34
Porsche						
z-order curve	72.73	263.09	857.53	2 492.10	7 142.92	22 457.98
Hilbert curve	77.91	281.10	919.52	2 714.14	7 892.19	25 043.80

Table 7. Memory consumption for PSVDAG building and encoding

bel/caller with at least 6 b binary representation. As the number of labels/callers in the data structure increases, this binary representation length gradually increases. Thus, the shortest ones represent a 5.33-fold compression rate, or, in other words: occupy only 18.75 % of the space.

- Minimization of the label/caller binary representation length is supported by separating labels assignments at every level of the graph, instead of assigning labels globally within the entire data structure. Thus, the label assignment starts at every level of the data structure with a length of the 6 b.
- Frequency-based compaction sorts the subgraph root nodes separately at each level l of the graph, according to their frequency of referencing from the level $l-1$. The process sorts labels/callers from the most frequently referenced nodes to the least frequently referenced ones. Individual root nodes of the template subgraphs are then assigned by labels, in the order from those having the shortest binary representation (6 b – attached to the most frequently referenced nodes) to those having the most extended binary representation (assigned to the least often referenced subgraph root nodes). It ensures the lowest possible number of

Time [s]	**Resolution**					
	128^3	256^3	512^3	$1K^3$	$2K^3$	$4K^3$
Angel Lucy						
z-order curve	0.021	0.032	0.079	0.196	0.402	1.279
Hilbert curve	0.021	0.033	0.081	0.205	0.423	1.331
Skull						
z-order curve	0.068	0.091	0.202	0.480	1.051	4.065
Hilbert curve	0.068	0.092	0.205	0.486	1.065	4.154
Porsche						
z-order curve	0.039	0.057	0.136	0.339	0.765	2.744
Hilbert curve	0.041	0.061	0.144	0.362	0.813	2.912

Table 8. Time consumption for PSVDAG building and encoding

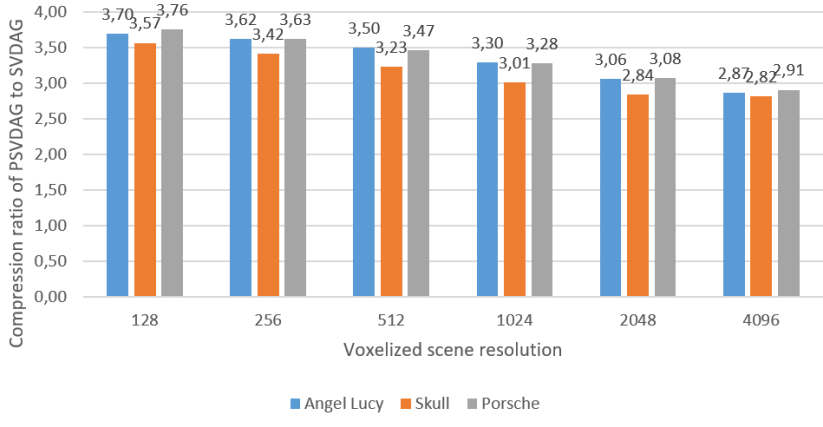


Figure 13. The PSVDAG-SVDAG compression ratio for each scene and the resolutions used to voxelize the respective scenes, when z-order applied

bits for the encoding of labels/callers per level of the graph.

6 CONCLUSIONS

This paper deals with hierarchical data structures designed to represent the geometry of voxelized three-dimensional scenes. It includes an overview of the related works published in the field of linearization of multi-dimensional data and the representation of two and three-dimensional data using dedicated data structures. The brief presentation of pointerless Sparse Voxel Octrees (SVO) and Sparse Voxel Directed Acyclic Graphs (SVDAG) summarized their advantages, disadvantages, and binary-level encoding. While pointerless SVO is a compact, pointerless data structure that is suitable for archival or streaming purposes, it is necessary to create

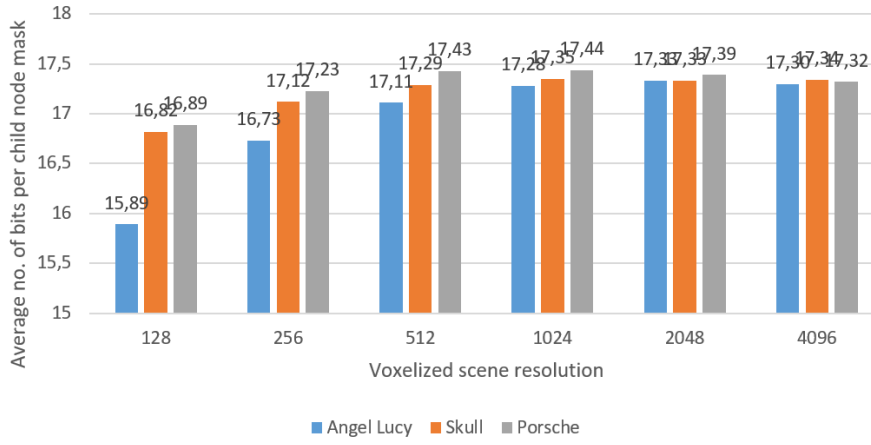


Figure 14. Average Child Node Mask size in bits for the scenes and the resolutions used to voxelize the respective scenes, when z-order applied

child node pointers for traversal. SVDAG is a more advanced data structure that incorporates 32 b child node pointers to allow fast traversal while reducing common subtrees.

In the subsequent section, the authors presented the proposed Pointerless Sparse Voxel Directed Acyclic Graph (PSVDAG) data structure, which combines the advantages of both data structures mentioned above. It allows a compact encoding

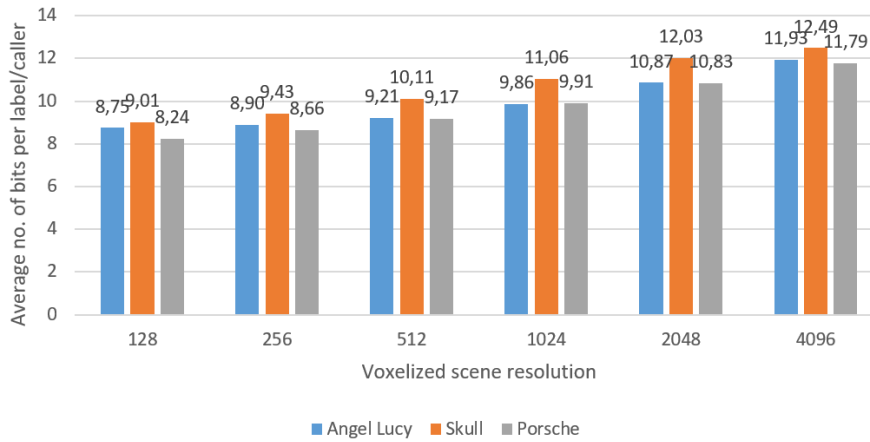


Figure 15. Average label/caller size in bits for the scenes and the resolutions used to voxelize the respective scenes, when z-order applied

of the data, and merging of common subtrees using the proposed concept of labels and callers having a variable-length binary representation. Compared to SVDAG, the disadvantage is that for fast traversal, this data structure requires the reconstruction of pointers to child nodes. The Active Child Node Count proposed in this paper facilitates and speeds up this process. In a favorable but less frequent case, it enables child node mask compression.

Performed tests were using scenes initially represented in the OBJ file format, and subsequently voxelized at various resolutions. The obtained results showed that – compared to the SVDAG data structure – PSVDAG achieved a compression ratio of 3.77 to 2.80 times higher. In most cases, PSVDAG outperformed the SVO, when the size of PSVDAG ranged from 131.1 % of the size of SVO (in the most unfavorable case) to 54.5 % of the size of SVO (in the most favorable case). Thus, in favorable circumstances, the binary representation of PSVDAG was more compact than that of SVO.

Acknowledgements

This research was supported by the Slovak Research and Development Agency, project number APVV-18-0214 and the KEGA Agency of the Ministry of Education, Science, Research and Sport of the Slovak Republic under Grant No. 003TUKE-4/2017 Implementation of Modern Methods and Education Forms in the Area of Security of Information and Communication Technologies towards Requirements of Labor Market.

REFERENCES

[1] Crassin, C.—Neyret, F.—Lefebvre, S.—Eismann, E.: GigaVoxels: Ray-Guided Streaming for Efficient and Detailed Voxel Rendering. Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games (I3D '09), ACM, 2009, pp. 15–22, doi: 10.1145/1507149.1507152.

[2] Balsa Rodríguez, M.—Gobetti, E.—Iglesias Guitián, J. A.—Makhinya, M.—Marton, F.—Pajarola, R.—Suter, S. K.: State-of-the-Art in Compressed GPU-Based Direct Volume Rendering. Computer Graphics Forum, Vol. 33, 2014, No. 6, pp. 77–100, doi: 10.1111/cgf.12280.

[3] Katajainen, J.—Mäkinen, E.: Tree Compression and Optimization with Applications. International Journal of Foundations of Computer Science, Vol. 1, 1990, No. 4, pp. 425–447, doi: 10.1142/S0129054190000291.

[4] Laszloffy, A.—Long, J.—Patra, A. K.: Simple Data Management, Scheduling and Solution Strategies for Managing the Irregularities in Parallel Adaptive *hp* Finite Element Simulations. Parallel Computing, Vol. 26, 2000, pp. 1765–1788, doi: 10.1016/S0167-8191(00)00054-5.

[5] Sagan, H.: Space-Filling Curves. Springer-Verlag, New York, 1994, doi: 10.1007/978-1-4612-0871-6.

[6] Hilbert, D.: Über die Stetige Abbildung Einer Linie auf ein Flächenstück. Mathematische Annalen, Vol. 38, 1891, pp. 459–460, doi: 10.1007/BF01199431.

[7] Morton, G. M.: A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing. Research Report. International Business Machines Corporation (IBM), Ottawa, Canada, 1966.

[8] Gargantini, I.: An Effective Way to Represent Quadtrees. Communications of the ACM, Vol. 25, 1982, No. 12, pp. 905–910, doi: 10.1145/358728.358741.

[9] Hunter, G. M.—Steiglitz, K.: Operations on Images Using Quad Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-1, 1979, No. 2, pp. 145–153, doi: 10.1109/TPAMI.1979.4766900.

[10] Klinger, A.—Dyer, C. R.: Experiments in Picture Representation Using Regular Decomposition. Computer Graphics and Image Processing, Vol. 5, 1976, No. 1, pp. 68–105, doi: 10.1016/S0146-664X(76)80006-8.

[11] Kawaguchi, E.—Endo, T.: On a Method of Binary-Picture Representation and Its Application to Data Compression. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-2, 1980, No. 1, pp. 27–35, doi: 10.1109/TPAMI.1980.4766967.

[12] Webber, R. E—Dillencourt, M. B.: Compressing Quadtrees via Common Subtree Merging. Pattern Recognition Letters, Vol. 9, 1989, No. 3, pp. 193–200, doi: 10.1016/0167-8655(89)90054-8.

[13] Chang, H. K.-C.—Liu, S.-H.—Tso, C.-K.: Two-Dimensional Template-Based Encoding for Linear Quadtree Representation. Photogrammetric Engineering and Remote Sensing, Vol. 63, 1997, No. 11, pp. 1275–1282.

[14] Parker, E.—Udeshi, T.: Exploiting Self-Similarity in Geometry for Voxel Based Solid Modeling. Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications (SM '03), 2003, pp. 157–166, doi: 10.1145/781606.781631.

[15] Srihari, S. N.: Representation of Three Dimensional Digital Images. Technical Report No. 162. Department of Computer Science, State University of New York at Bufallo, Amherst, New York, pp. 26, 1980.

[16] Rubin, S. M.—Whitted, T.: A 3-Dimensional Representation for Fast Rendering of Complex Scenes. Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '80), ACM, 1980, pp. 110–116, doi: 10.1145/800250.807479.

[17] Jackins, C. L.—Tanimoto, S. L.: Oct-Trees and Their Use in Representing Three-Dimensional Objects. Computer Graphics and Image Processing, Vol. 14, 1980, No. 3, pp. 249–270, doi: 10.1016/0146-664X(80)90055-6.

[18] Meagher, D. J. R.: Octree Encoding: A New Technique for the Representation, Manipulation, and Display of Arbitrary 3-D Objects by Computer. Technical Report No. IPL-TR-80-111. Rensselaer Polytechnic Institute, Troy, NY, 1980.

[19] Meagher, D.: Geometric Modeling Using Octree Encoding. Computer Graphics and Image Processing, Vol. 19, 1982, No. 2, pp. 129–147, doi: 10.1016/0146-664X(82)90104-6.

[20] Meagher, D. J. R.: The Octree Encoding Method for Efficient Solid Modeling. Technical Report IPL-TR-032, Image Processing Laboratory, Rensselaer Polytechnic Institute, Troy, New York, 1982.

[21] Samet, H.: Implementing Raytracing with Octrees and Neighbor Finding. Computers and Graphics, Vol. 13, 1989, No. 4, pp. 445–460, doi: 10.1016/0097-8493(89)90006-X.

[22] Knoll, A.—Wald, I.—Parker, S. G.—Hansen, C. D.: Interactive Isosurface Ray Tracing of Large Octree Volumes. 2006 IEEE Symposium on Interactive Ray Tracing, Salt Lake City, UT, USA, 2006, pp. 115–124, doi: 10.1109/RT.2006.280222.

[23] Knoll, A. M.—Wald, I.—Hansen, C. D.: Coherent Multiresolution Isosurface Ray Tracing. The Visual Computer, Vol. 25, 2009, No. 3, pp. 209–225, doi: 10.1007/s00371-008-0215-2.

[24] Schnabel, R.—Klein R.: Octree-Based Point Cloud Compression. Proceedings of the 3rd Eurographics Symposium on Point-Based Graphics/IEEE VGTC Conference on Point-Based Graphics (SPBG '06), 2006, pp. 111–121, doi: 10.2312/SPBG/SPBG06/111-120.

[25] Laine, S.—Karras, T.: Efficient Sparse Voxel Octrees. Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '10), ACM, 2010, pp. 55–63, doi: 10.1145/1730804.1730814.

[26] Kämpe, V.—Sintorn, E.—Assarsson, U.: High Resolution Sparse Voxel DAGs. ACM Transactions on Graphics, Vol. 32, 2013, No. 4, Art. No. 101, 8 pp., doi: 10.1145/2461912.2462024.

[27] Villanueva, A. J.—Marton, F.—Gobbetti, E.: SSVDAGs: Symmetry-Aware Sparse Voxel DAGs. Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D '16), ACM, 2016, pp. 7–14, doi: 10.1145/2856400.2856420.

[28] Čerešník, P.—Madoš, B.—Baláž, A.—Bilanová, Z.: SSVDAG*: Efficient Volume Data Representation Using Enhanced Symmetry-Aware Sparse Voxel Directed Acyclic Graph. Proceedings of the 2019 IEEE 15th International Scientific Conference on Informatics, IEEE, 2019, pp. 70–75, doi: 10.1109/Informatics47936.2019.9119298.

[29] Baert, J.—Lagae, A.—Dutré, P.: Out-of-Core Construction of Sparse Voxel Octrees. Proceedings of the 5th High-Performance Graphics Conference (HPG '13), ACM, 2013, pp. 27–32, doi: 10.1145/2492045.2492048.

Liberios Vokorokos graduated (M.Sc.) with honors at the Department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice in 1991. He defended his Ph.D. in the field of programming devices and systems in 2000 with the thesis title "Diagnosis of Compound Systems Using the Data Flow Applications". He serves as Professor for computer science and informatics since 2005. Since 1995 he is working as Educationist at the Department of Computers and Informatics. His scientific research focuses on parallel computers of the data flow type. He also investigates the questions related to the diagnostics of complex systems. Currently, he is the Dean of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice. He is a member of the Advisory Committee for Informatization at the Faculty and Advisory Board for the Development and Informatization at the Technical University of Košice.

Branislav Madoš graduated (Ing.) at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice in 2006. He defended his Ph.D. in the field of computers and computer systems in 2009. His thesis title was "Specialized Architecture of Data Flow Computer". Since 2010 he is Assistant Professor at the Department of Computers and Informatics. His scientific research focuses on parallel computer architectures and architectures of computers with a data-driven computational model, and on computer security using cryptographic and steganographic methods.

Zuzana Bilanová graduated (Ing.) at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice in 2015. Since 2015 she is Ph.D. student at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. Her main scientific focus is creating new approaches in logical analysis of natural language, non-classical logical systems in computer science, and resource-oriented logic programming. Her secondary research areas are educational technologies for the effective implementation of engineering education, focusing on project and team-based teaching.