Speedup of Parsing for Recognition of Online Handwritten Mathematical Expressions

Anh Duc Le and Masaki Nakagawa
Department of Computer Information
Tokyo University of Agriculture and Technology
Email: leducanh841988@gmail.com, nakagawa@cc.tuat.ac.jp

***Abstract*—This paper proposes a method for speeding up parsing process for recognizing online handwritten mathematical expressions (OHME). We prune infeasible partitions in the parsing table to reduce the time for the parsing process. Low score partitions are candidates for pruning. Our method can be applied for any parsing algorithms that use score functions. In this paper, we use a stroke order free system as a baseline system. The method is as follows. First, we analyze the scores of partitions in each row of the parsing table. Then, we determine a threshold for each row to prune low score partitions. Finally, we employ these thresholds to prune low score partitions on the baseline recognition system. The results of evaluations of our method on the CROHME 2014 database show that the recognition process is speeded up by 3.46 times and 4.97 times while recognition rate is reduced only 0.31 point and 0.71 point, respectively.**

Keywords—online handwritten mathematical expressions, handwriting recognition, parsing algorithm, speedup.

I. INTRODUCTION

Recognition of handwritten mathematical expressions is one of the current challenges concerning handwriting recognition. It could be used in many applications such as an input method for equations, tutoring systems, and so on. Educational applications employ recognition of OHMEs to develop self-learning and automatic marking systems. Students input answers by handwriting and receive feedback to their answers immediately. The recognition rate and processing time are the two most important factors to apply it to real applications. Currently, researchers focus on improving recognition rate, however, processing time is also important.

Recognition of OHMEs can be divided into three main processes. First, a sequence of input strokes is segmented into hypothetical symbols (symbol segmentation), where each stroke is a sequence of coordinates from pen/touch-down to pen/touch-up. Second, each hypothetical symbol is then recognized by a symbol classifier (symbol recognition). Finally, structural relations among the recognized symbols are determined and the structure of an OHME is analyzed by parsing the recognized symbols and relations to determine the most likely interpretation as an OHME (structural analysis). The parsing process is the most difficult process because it takes almost all processing time for searching for the best result. If we reduce its processing time, the OHME recognition will become more practical.

Recognition of OHMEs has been studied since around 1970 and become an active research topic recently. Many approaches have been proposed for recognizing OHMEs especially during the last two decades. They are summarized in the survey papers [1, 2] and the recent competition papers [3, 4]. We will review a few recent approaches that participated in the recently competition on recognition of OHMEs (CROHME 2014).

Alvaro et al. proposed a formal model for OHME recognition based on 2 dimensional Stochastic Context Free Grammar (SCFG) and combination of RNN classifiers for online and offline features [5, 6]. The parsing table is modified into one index to parse an input OHME in two dimensions. For combining two sub-partitions, the structural relation between two sub-partitions are used instead of stroke order, since their system is independent from stroke order. However, the complexity of parsing is increased. Although the range search is employed for optimizing the complexity, the complexity of the parsing algorithm is quite large as $O(n^3 ln(n)|P|)$. We abbreviate stroke order free methods like this as SOF.

Le et al. formulated the recognition problem as a search problem of the most-likely OHME candidate in a framework of SCFG [7, 8]. Stroke order is employed to reduce the search space and the Cocke-Younger-Kasami (CYK) algorithm is employed to parse a sequence of input strokes. Therefore, the complexity of the parsing algorithm is still $O(n^3|P|)$, like that of the original CYK algorithm. The grammar rules were extended to handle multiple symbol order variations. They proposed a concept of body box and two Support Vector Machine (SVM) models for classifying structural relations. The experiments show that the recognition rate is improved and the processing time is practical. We shorten stroke order dependent methods such as this as SOD. They also proposed a reordering method to transform a SOD system to a SOF system [9].

A global approach allowing mathematical symbols and structural relations to be learned directly from expressions was proposed by Mouchere et al. [10]. During the training phase, symbol hypotheses are generated without using a language model. A dynamic programming algorithm finds the best segmentation and recognition of the input. A classifier learns both the correct and incorrect segmentations. The training process is repeated to update the classifier until the classifier recognizes the training set of OHMEs correctly. Furthermore, contextual modeling based on structural analysis of OHMEs is employed, where the models are learnt directly from OHMEs using the global learning scheme.

A modified version of the MST-based parsing algorithm was presented by Davila et al. [11]. The parser recursively groups vertical structures (e.g. fractions, summations and square roots), extracts the dominant operator (e.g. fraction line) in each vertical group, and then detects symbols on the main baseline and on baselines in superscripted and subscripted regions. For each baseline, an MST is built over symbol candidates and the spatial relations between symbols. Spatial relations are classified using bounding box geometry and shape context features for regions around symbol pairs [12]

A parsing algorithm composed of two stages was presented by Yao et al. [4]. For the first parsing stage, strong geometric relations (upper, lower and inside) are detected. To solve the upper/lower spatial relationships, every fraction bar and its upper and lower expressions are detected. Then, root symbols and contained expressions are detected. For the second parsing stage, Shape Feature Polar Histograms and SVM are used to detect weak geometric relations (horizontal, superscript and subscript).

A method based on baseline tree extraction and syntactical parsing was presented by Aguilar et al. [4]. First, symbol hypotheses are generated by combining neighbor strokes. Next, each symbol hypothesis is recognized by a symbol classifier with a reject option. Then, several partitions, using the symbol hypothesis with the lowest cost, is generated and a baseline structure tree is built for each partition. Finally, the latex code of each baseline tree is extracted and a parser evaluates if it belongs to a predefined mathematical language grammar. The legal expression with lowest cost is selected for output.

A lot of researches in recent years have focused on improving recognition rate for recognition of OHMEs as mentioned above. However, few studies have focused on reducing the processing time of the recognition system. The work by Le et. al. compares the SOF and SOD parsing algorithms on recognition rate, memory space usage, and processing time [12]. However, they did not propose any method for reducing the processing time.

In this paper, we propose a method to reduce the processing time of a parsing algorithm. In this work, it is applied for the SOF parsing algorithm. However, it can be applied for any parsing algorithms that use score functions for parsing. The procedure of the proposed method is as follows. First, we train a SOF system as a baseline system. For reducing the processing time, we analyze the scores of cells in the parsing table to remove infeasible partitions. Second, we create a set of thresholds to remove infeasible cells. Finally, the thresholds are integrated to the parsing algorithm to remove cells with low scores.

The rest of this paper is organized as follows. The baseline system for recognition of OHMEs is presented in Section 2. The proposed method for reducing the processing time of the parsing algorithm is described in Section 3. The results of an evaluation of the proposed method are presented and discussed in Section 4. Conclusions are drawn in Section 5.

II. OVERVIEW OF THE RECOGNITION SYSTEM

The problem of OHME recognition is formulated as a search problem of the most-likely interpretation of handwritten strokes. The search problem is modeled as the following score function for a candidate expression of n symbols connected by m relations:

$$C = \alpha_1 \sum_{i=1}^{n} \ln(P_{sh}(\mathrm{G}_i)) + \alpha \sum_{i=1}^{n} \ln(P_{rec}(\mathrm{S}_i|\mathrm{G}_i)) + \alpha_3 \sum_{k=1}^{m} \ln(P_{\mathrm{rel}}(\mathrm{R}_k|\mathrm{B}_k\mathrm{C}_k)) + \alpha_4 \sum_{k=1}^{m} \ln\left(P_{Gram}(\mathrm{A}_k \xrightarrow{\mathrm{R}_k} \mathrm{B}_k\mathrm{C}_k)\right) \quad (1)$$

where G_i is a symbol hypothesis composed of set of strokes; $P_{sh}(G_i)$ stands for the probability of a symbol hypothesis G_i; $P_{rec}(S_i|G_i)$ stands for the probability that a symbol hypothesis G_i is recognized as a symbol S_i; $P_{rel}(R_k|B_kC_k)$ is the probability that two sub-expressions B_k and C_k are combined into a larger expression with a relation R_k, and $P_{Gram}(\mathrm{A}_k \xrightarrow{\mathrm{R}_k} \mathrm{B}_k\mathrm{C}_k)$ is the probability of a production $\mathrm{A}_k \xrightarrow{\mathrm{R}_k} \mathrm{B}_k\mathrm{C}_k$ in the grammar. The coefficients α_1, α_2, α_3, α_4 are the weighting parameters for probabilities. These parameters are trained by Genetic Algorithm on a validation set. The description of probabilities is reviewed in the remaining of this section.

A. Probability of symbol hypothesis

Probability of a symbol hypothesis $P_{sh}(G_i)$ is calculated from a set of strokes: horizontal and vertical projections of center-to-center distance of bounding boxes, stroke size difference (stroke size is the larger of height and width), and the minimum pairwise distance among all the strokes. We use a Gausian Mixture Model classifier for obtaining this probability.

B. Probability of symbol recognition

The combined recognizer composed of offline and online recognition methods is robust due to the advantages of both the methods. Particularly, it could recognize connected strokes or cursive strokes by the online method and stroke disorders or duplicated strokes by the offline method.

Bidirectional long short term memory Neural Network (BLSTM) and Convolution Neural Network (CNN) are used for online and offline symbol classification, respectively. Online features and local gradient features are employed to improve accuracy of BLSTM. The maxout nonlinearity and the dropout technique are employed to improve the performance of CNN. The combination equation is shown in (2), where β is the weighting parameter to balancing the contributions of BLSTM and CNN. The detail of this combined classifier is reported in [13].

$$\mathrm{P}_{\mathrm{rec}}(\mathrm{S_i}|\mathrm{G_i}) = \beta\, P_{BLSTM}(\mathrm{S_i}|\mathrm{G_i}) + (1-\beta)\, P_{CNN}(\mathrm{S_i}|\mathrm{G_i})$$

C. Probability of structural relation

For determination of structural relation, we employ a concept of body box instead of bounding box for symbols/MEs [8]. Four features of D_x, D_y, H and O are extracted as shown in Figure 1. The feature D_x shows the relation between the horizontal centers of 2 body boxes for symbol/sub-expressions, to divide above, below, inside relations into Group 1 and horizontal, superscript, subscript relations into Group 2. Then, the features H and D_y classify the above, below, inside relations in Group 1 by a SVM while the H, D_y and O features classify the horizontal, superscript, subscript relations in Group 2 by another SVM. $P_{rel}(R_k|B_kC_k)$ is calculated by transforming a score of SVM to a probability.

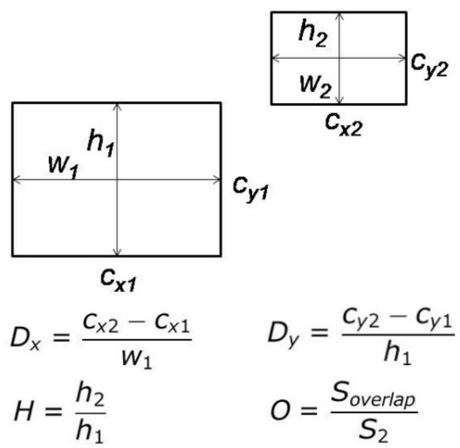

Figure 1.Features used for determining structural relations

D. *Probability of grammar rule*

A SCFG is defined formally by a five-tuple $G = (N, \Sigma, R, P, S)$ where:

- N is a finite set of non-terminal symbols.
- Σ is a finite set of terminal symbols.
- R is a finite set of relations between 2 sub-expressions. These relations are horizontal, over, under, superscript, subscript, inside.
- P is a finite set of productions of one of the following forms: $X \xrightarrow{r} A\ B$. With $X, A, B \in N$, $r \in R$. Each grammar rule is associated with a probability p and these probabilities satisfy the condition: $\forall X \in N : \sum p(X \to \alpha) = 1$. These probabilities are trained by the Viterbi algorithm.
- $S \in N$ is a distinguished start symbol.

E. *Parsing Algorithm*

The parsing processing contains two stages: initial stage and parsing stage.

Initial stage: The parsing table is initialized by the symbol hypotheses described in section 2.A. Their probability is calculated by Eq. (2).

$$C_{init} = \alpha_1 \ln(P_{sh}(G_i)) + \alpha\ \ \ln(P_{rec}(\alpha|G_i)) + \alpha_4 \ln\left(P_{Gram}(A_k \to \alpha)\right) \quad (2)$$

Parsing stage: To create a sub-expression z under the grammar rule ($X \xrightarrow{r} A\ B$), we look up two sub-partitions x and y, whose structural relation satisfies the relation r and $|z| = |x| + |y|$. Each partition is stored in a cell of the parsing table. The score of the combination of sub-partitions P_1 and P_2 under the grammar rule ($X \xrightarrow{r} A\ B$) is calculated by Eq. (3).

$$C = C(P_1, A) + C(P_2, B) + \alpha_3 \ln(P_{rel}(R_k|P_1P_2)) + \alpha_4 \ln\left(P_{Gram}(A_k \xrightarrow{R_k} B_k C_k)\right) \quad (3)$$

where $C(P_1, A)$ and $C(P_2, B)$ are scores of partitions P_1 and P_2 that are recognized as non-terminal symbols A and B, respectively.

Figure 2 presents search regions of sub-partitions according to relations. This is similar to the definition of search regions by Alvaro et al. Although the search regions prune many infeasible combinations, the number of cells of the parsing table is still larger. Even an optimization is applied by the binary search, the complexity of the parsing algorithm is still $O(n^3 ln(n)|P|)$. The pruning of infeasible combinations is an interesting problem for improving the recognition rate and the processing time.

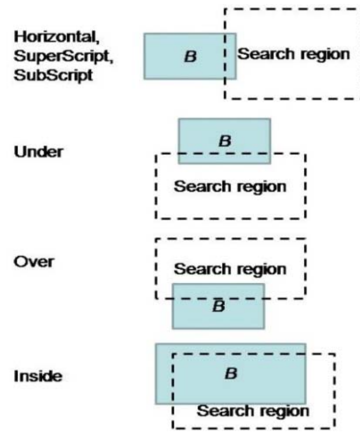

Figure 2. Search regions of a sub-partition (B) according to relations.

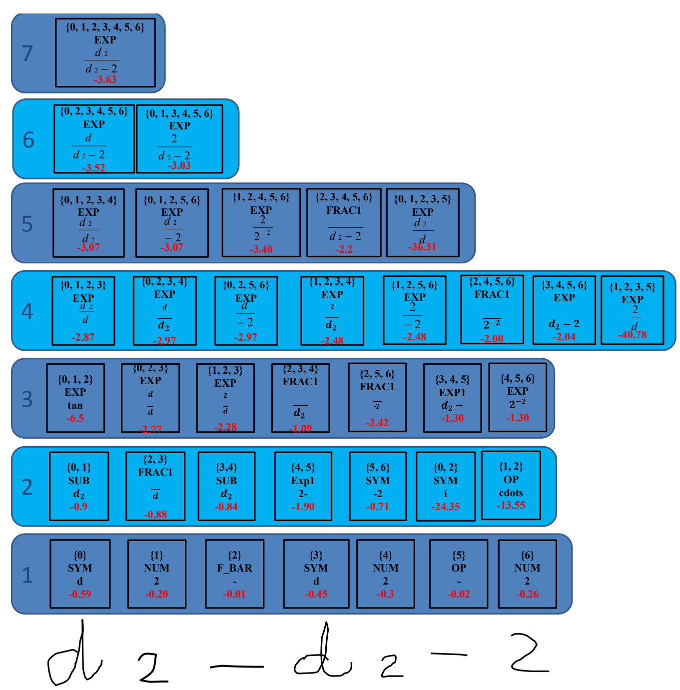

Figure 3. Result of the SOF parsing table for the OHME $\frac{d_2}{d_2 - 2}$.

Figure 3 shows an example of the parsing table for the OHME $\frac{x^2}{2x}$. In each cell, we show only the best candidate which contains the group of stroke indexes, the non-terminal symbol, the result and its score. The total number of cells in the parsing table increases rapidly, because there are many combination in two dimensions. The score of each cell contains an important information whether it is a feasible or infeasible sub-expression. The larger score a cell has, the more confidence that the cell can be a feasible sub-expression.

III. Speeding up method for parsing

The basic idea of our method is to prune infeasible partitions in the parsing table. Partitions having low scores are good candidates for pruning. We run the baseline recognition system on a validation set to get scores of partitions. Then, we analyse the scores base on the size of partitions. Next, we determine a threshold for each size of partition to prune partitions having low scores. Finally, we employ these thresholds to prune partitions having low scores in the baseline recognition system.

A. Analyzing scores of partitions and determining a set of thresholds

Figure 4 shows the cumulative distributions of scores for partitions whose lengths are from 1 to 10. From the distribution, we can draw a line in the cumulative percent axis to find a set of scores and prune partitions by cutting the distribution by the line. Depending on how many percent that we want to prune, we can determine the corresponding threshold. In Figure 4, two dotted lines represent 50% and 75% pruning. The scores to reduce 50% and 75% of partitions of the length 1 are -7.72 and -3.03, respectively. Table 1 shows the thresholds for pruning 50% and 75% of partitions whose lengths are from 1 to 10. The thresholds increase when lengths of partitions increase. We determine the threshold for the length from 1 to 40.

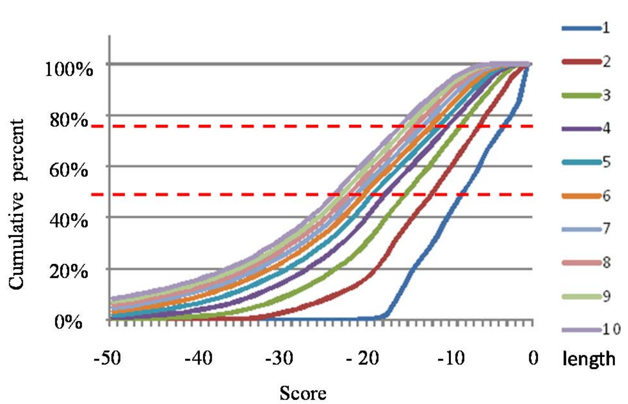

Figure 4. The cumulative distributions of scores for partitions whose lengths are from 1 to 10.

TABLE I. THE THRESHOLD FOR PRUNING 50% AND 75% OF PARTITIONS WHICH LENGTH IS FROM 1 TO 10.

Length of partition	**The percent of partitions to be pruned**	
	50%	75%
1	-7.73	-3.-3
2	-9.94	-4.95
3	-11.89	-6.35
4	-13.58	-7.47
5	-14.91	-8.39
6	-15.97	-9.19
7	-16.84	-9.92
8	-17.65	-10.57
9	-18.38	-11.19
10	-19.05	-11.77

B. Apply to the parsing algorithm

Finally, we use the thresholds determined above to prune partitions having lower scores in the parsing process. We prune infeasible partitions whose lengths are equal or less than 40. Figure 5 shows the parsing table after pruning 50% of partitions having lower scores. It has less cells than the previous parsing table in Figure 3.

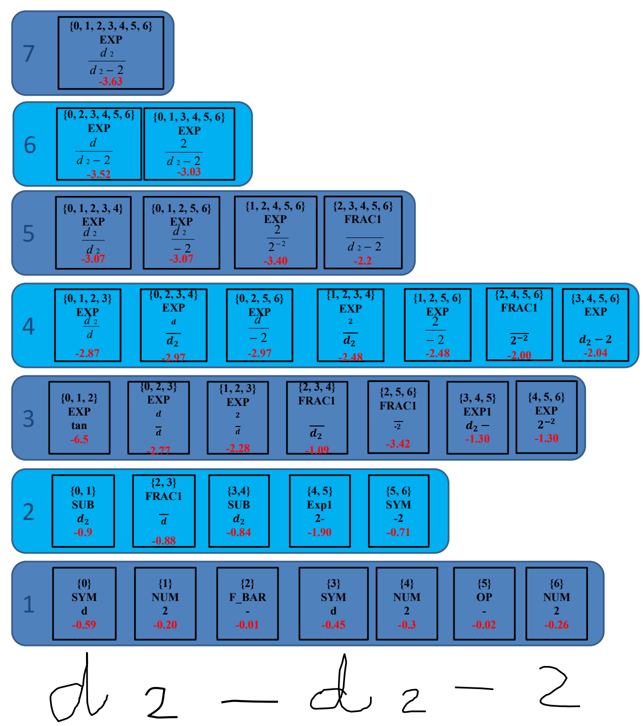

Figure 5. Result of the SOF parsing table after pruning 50% of partitions having low scores.

IV. Evaluation

A. Databases

We use the CROHME 2014 database [11]. Organized at ICHFR 2014, CROHME 2014 was a contest in which OHME recognition algorithms competed. It allowed the performance of the proposed system to be compared with others under the same conditions. There were seven participants. The CROHME 2014 database contains 8,836 OHMEs for training and 986 OHMEs for testing. The number of symbol classes is 101, including many similar symbols such as {*1*, |, *l* }, {*P*, *p*}, {*S*, *s*}, {*C*, *c*}, {*X*, *x*, ×},{*V*, *v*}, and {*o*, *0*}. We employed the CROHME 2013 test set as the validation set. The validation set was used for estimating parameters, analyzing probabilities, and determining thresholds for the recognition systems.

All the experiments were performed on an Intel(R) Core(T7) i7 2.93-GHz CPU with 4.0-GB memory.

B. Experiments

In this experiment, the percent of partitions is reduced from 50 to 90 with the step 10. We call these systems as P_{50}, P_{60}, P_{70}, P_{80}, P_{90}. The first experiment evaluated the performance of the baseline system, P_{50}, P_{60}, P_{70}, P_{80}, P_{90} on the CROHME 2014 database in order to compare them with the other systems participated in the CROHME 2014 competition and stroke order dependent (SOD) system by Le et al. [12]. The four factors were measured in the evaluation, namely, *Sym Seg* as symbol segmentation rate, *Sym Seg + Rec* as symbol segmentation and recognition rate, *Rel Tree* as rate of structural analysis (termed "*relation tree*"), and *Exp Rec* as expression recognition rate as listed in Table II. The Baseline system is slightly superior to the best system using only the CROHME training set (System I) in recognition rate. Recognition rate of the proposed systems is from 4.36% to 37.22%. It drops less than 1 point from the baseline system when we prune from 50% to 70% of partitions. When we prune 80% of partitions, it drops 4.5 point. When we prune 90% of partitions, it significantly drops.

TABLE II. RECOGNITION PERFORMANCE ON CROHME 2014 (%).

System	**Measure**			
	Sym Seg	**Seg+Class**	**Rel Tree**	**Exp Rec**
I	93.31	86.59	84.23	37.22
II	76.63	66.97	60.31	15.01
III	98.42	93.91	94.26	62.68
IV	85.52	76.64	70.78	18.97
V	88.23	78.45	61.38	18.97
VI	83.05	69.72	66.83	25.66
VII	89.43	76.53	71.77	26.06
SOD system	89.85	83.21	71.55	35.80
Baseline system	90.82	84.04	80.58	37.53
***P_{50}* system**	82.68	76.81	72.80	37.22
***P_{60}* system**	82.55	76.79	72.74	37. 12
***P_{70}* system**	81.59	76.22	71.71	36.82
***P_{80}* system**	77.20	72.83	66.39	33.06
***P_{90}* system**	34.21	32.51	21.48	4.36

The second experiment evaluated processing time and complexity of the parsing algorithms. For the complexity of the parsing algorithms, we counted the number of cells in the parsing table (the number of sub-partitions generated on the parsing process). The average stroke number per OHME is 13.99. Figure 6 and 7 show the average processing time and the number of cells in the parsing table with expression recognition rate. The average time is speeded up from 3.46 times to 10.29 times while the number of cells was reduced from 1.38 times to 21.97 times. If we consider both recognition rate and processing time, we should prune partitions from 50% to 70%.

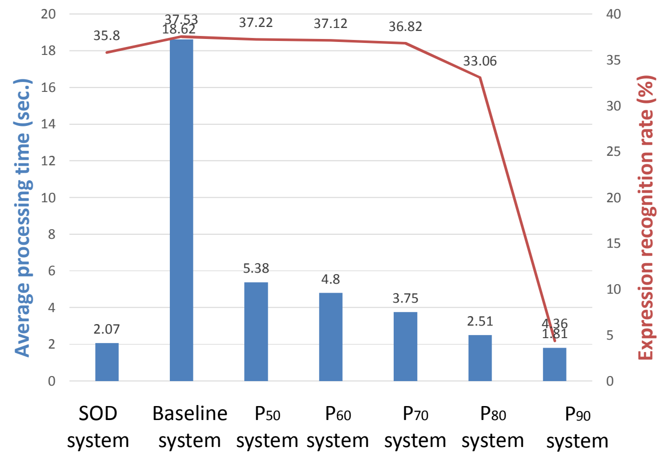

Figure 6. Comparison of average processing time.

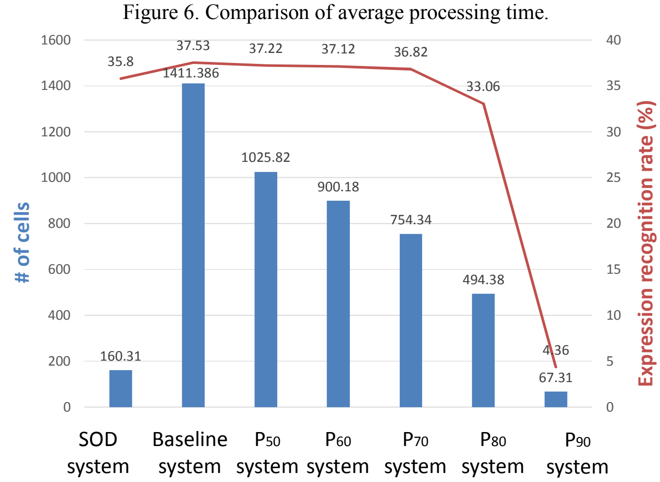

Figure 7. Comparison of the number of cells in the parsing table.

C. Future works

Currently, we are analyzing all scores of partitions in the parsing table. In the future, we should investigate how to prune infeasible partitions that do not affect to the recognition rate. We should use not only scores of partitions but also other information to determine infeasible partitions.

V. CONCLUSION

In this paper, we presented a method for pruning infeasible partitions in order to speed up the parsing process. The method estimates infeasible partitions by their scores. Low score partitions are candidates for pruning. We analyzed scores of partitions and determined the thresholds for pruning. Then, we employed these thresholds to prune partitions that have scores less than the corresponding threshold. The method is simple but effective for speeding up the parsing process. Though the experiments on the CROHME 2014 database, we have observed that the percent of pruned partitions should be from 50% to 70%. Our system speeds up 3.46 times and 4.97 times of processing while dropping 0.31 point and 0.71 point of recognition rate, respectively.

There remains a problem of pruning more infeasible partitions effectively by using more information.

ACKNOWLEDGEMENT

This research has been supported by JSPS fellowship under the number 15J08654. The authors thank Dr. Alvaro for helpful comments and discussions and availing the parsing algorithm.

REFERENCES

[1] K. Chan and D. Yeung, Mathematical Expression Recognition: A Survey, International Journal of Document Analysis and Recognition, pp. 3-15, 2000

[2] R. Zanibbi and D. Blostein, Recognition and Retrieval of Mathematical Expressions, International Journal of Document Analysis and Recognition , pp.331-357, 2012.

[3] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, U. Garain, D.H. Kim and J.H. Kim, ICDAR 2013 CROHME: Third International Competition on Recognition of Online Handwritten Mathematical Expressions. Proc. Int'l Conf. Document Analysis and Recognition, pp.1428-1432, Washington, DC.

[4] H. Mouchere, C. Viard-Gaudin,R. Zanibbi, and U. Garain, ICFHR 2014 Competition on Recognition of On-line Handwritten Mathematical Expressions (CROHME 2014). Proc. Int'l Conf. Frontiers in Handwriting Recognition, pp. 791-796, 2014.

[5] F. Álvaro, J. Sánchez, and J. Benedí, Recognition of On-line Handwritten Mathematical Expressions Using 2D Stochastic Context-Free Grammars and Hidden Markov Models, Pattern Recognition Letters, pp. 58-67, 2014.

[6] F. Alvaro, J. Sánchez, and J. Benedí, An integrated grammar based approach for mathematical expression recognition, Pattern Recognition, vol. 51, pp. 135–147, 2016

[7] A.D. Le, T.V. Phan, and M. Nakagawa, A System for Recognizing Online Handwritten Mathematical Expressions and Improvement of Structure Analysis, 11th IAPR Workshop on Document Analysis Systems, 2014, pp. 51-55.

[8] A.D. Le and M. Nakagawa, A system for recognizing online handwritten mathematical expressions by using improved structural analysis, International Journal of Document Analysis and Recognition, vol. 19, pp 305–319, 2016

[9] A. D. Le, H. D. Nguyen, M. Nakagawa, Modified XY Cut for Re-Ordering Strokes of Online Handwritten Mathematical Expressions, 12th IAPR Workshop on Document Analysis Systems, 2016, pp. 233-238.

[10] A. M. Awal, H. Mouchère, and C. Viard-Gaudin, A global learning approach for an online handwritten mathematical expression recognition system Pattern Recognition Letters, pp 68-70, 2014.

[11] L Hu, R Zanibbi, MST-based visual parsing of online handwritten mathematical expressions, 15th International Conference on Frontiers in Handwriting Recognition, 2016, pp. 337-342.

[12] A.D. Le and M. Nakagawa, Comparison of parsing algorithms for recognizing online handwritten mathematical expressions, 15th International Conference on Frontiers in Handwriting Recognition, 2016, pp. 390-394.

[13] H. D. Nguyen, A. D. Le, M. Nakagawa. Combination of LSTM and CNN on recognizing mathematical symbols, in Information-Based Induction Sciences and Machine Learning (IBISML2014), pp. 287-292, 2014.