Segmentation and graph matching for online analysis of student arithmetic operations

Arnaud Lods, Eric Anquetil, Sébastien Macé

Segmentation and graph matching for online analysis of student arithmetic operations*

Arnaud Lods[1,2] (✉)[0000−0002−2528−4586], Éric Anquetil[2][0000−0002−1760−5095], and Sébastien Macé[1]

[1] Learn&Go
{firstname.lastname}@learn-and-go.com
[2] Univ Rennes, CNRS, IRISA, F-35000 Rennes
{firstname.lastname}@irisa.fr

Abstract. This paper is based on a research project aiming at improving learning long arithmetic operations in primary school using pen-based tablets. The goal is to automatically analyze a student's handwritten answer by comparing it to an expected answer and to provide immediate feedback. This comes down to find any mistake made such as a calculus mistake, missing carry over or symbol misalignment. We use the correspondence obtained by the Graph Edit Distance (GED) computed between both the student and expected answers. In order to reduce graph sizes to overcome the computational complexity of the GED on large graphs, we present a new semantic graph of line segmentation. We propose a backtracking process to correct potential early mis-recognition mistakes for non-corresponding vertices. We evaluate the improvement on the analysis performances for an increasing number of backtracks on an in-house dataset composed of 400 handwritten operations.

Keywords: arithmetical operation analysis · graph edit-distance · online hand-drawn structured document recognition

1 Introduction

Through the use of Intelligent Tutors, we are able to provide students with immediate feedback to help them correct their mistakes. For mathematics some tutors were proposed using keyboard interfaces [1, 2], but with the improvement of pen-tablet devices such systems can be enhanced to transfer solving mathematical problem from numerical devices to paper back and forth. The on-line handwritten input is a list of sequences of points in the 2D space. A long arithmetic operation is a 2D structure represented by a set of related handwritten mathematical symbols. Figure 1 displays an example of such input. Given a long arithmetic problem we can represent the expected answer to confront it to the handwritten answer made by the student. The goal is to produce adapted

* With the support from the LabCom **ScriptAndLabs** founded by the **ANR** ANR-16-LVC2-0008-01. With the support from the ANRT.

feedback for the student, which depends on the nature of its mistakes. Given the learning context, mistakes can be a wrong instruction recopy, misaligned symbols, a forgetting of symbols, calculus mistakes, an excess of symbols... (see Figure 5) Due to the complexity of the input, there is a need to devise a recognition system to transform the handwriting into an adapted representation. The comparison between this representation and the expected answer also needs to be computed in no more than a couple of seconds to present "immediate" feedback to the students.

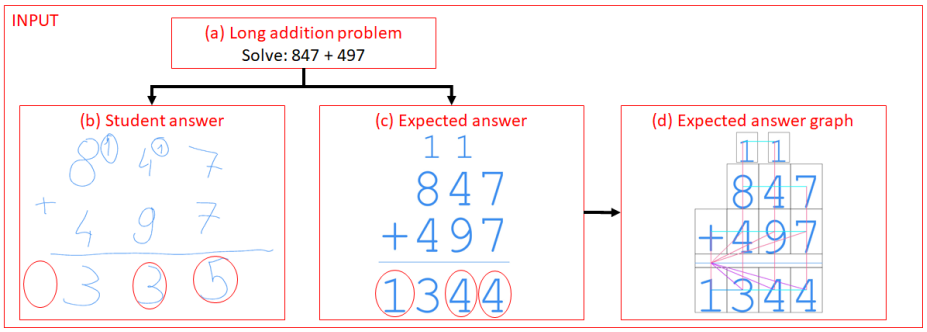

Fig. 1: Input of the system. (a) A long arithmetic problem is given to the student. (b) His answer is handwritten using a pen-tablet device. (c) An expected answer is created based on the problem. (d) A graph-based representation is generated. The dissimilarities we are looking for as an output of the analysis system are circled.

To represent both the handwritten input and the expected answer, we build a graph-based representation commonly used to represent mathematical expression and their symbols relationships [3]. We can produce corresponding representations to compute a comparison between the two graphs using the Graph edit Distance [4]. It is a popular and general graph similarity computation that searches the best vertices and edges correspondence between a pair of graphs. The dissimilarities found between graphs can be used to produce the analysis. For instance corresponding vertices with different labels might highlight calculus mistakes. However long arithmetic operation graphs easily reach 15+ vertices. To overcome the computational complexity of the GED on such graphs, we propose the use of a new graph segmentation. We transform the initial graph of symbols into a graph of lines by clustering symbols. This greatly reduces the size of graphs while keeping their structural identity. We propose to use the GED results by introducing backtracking steps into the system. Non-corresponding vertices between graphs can be re-evaluated to create several new recognition hypotheses to find a better fit to the expected answer.

1.1 Related works

Handwritten mathematical expression (HME) recognition is a widely researched subject with the popular CROHME competition [3]. The most recent and best performing systems use end-to-end neural network by transforming the set of online strokes to an image. A CNN is used to extract a features map encoded and decoded by a Recurrent Neural Network with attention based model [5]. Those systems are limited to the translation of an HME to a recognized expression, reducing the relationships between symbols to the bare minimum to construct the LaTeXexpression. Other works focus on sequential solutions to build graphs, such as visibility graphs [6, 7], representing mathematical symbols by vertices and relationships by edges. Those graphs are then parsed using a grammar to produce a Symbol Layout Tree containing only relationships to build a valid mathematical expression. The lower recognition results are mainly due to the missing context on the recognition step introduced by attention model in neural network. However they create a complete representation of the 2D structure of the mathematical expression in regards to relationships between symbols. In order to produce explicit feedback to the students, we need such an explicit representation of the structure of the operation with the complete relationships between symbols.

Given the graph representation of a student answer, it is then possible to compare it to another generated expected answer with the same architecture. The common method used for exact graph comparison are based on the Graph-Edit Distance, initially introduced in [8], which searches for the best pair-by-pair transformation from one graph to another. Several algorithms on the exact computation of the GED are presented in [4], and though several improvement using Depth-First Search [9], using sub-graph isomorphism [10] or through a mathematical solver [11] are proposed, it is yet challenging to compute this GED on graphs larger than 16 vertices without reaching the fixed time-out of 100 seconds. In practice we are able to compute the GED in a couple of seconds for smaller sizes of graphs. Other much faster methods rely on an inexact computation of the GED [12, 13]. Though efficient, an incorrect matching between symbols could induce unexpected dissimilarities between the student answer and the expected answer, and thus produce incorrect feedback. Other methods focus on graph sizes reduction by clustering graph nodes such as in [14]. In [15] they use this graph clustering algorithm to create hierarchical graph representation in order to embed graphs to a vectorial space while keeping most of the important structural information.

To overcome the computational complexity of the GED on larger graphs, we present a similar approach to graph clustering with an adapted graph segmentation for arithmetical operation to reduce the sizes of graphs while keeping the structural information of the operation. Furthermore, to avoid early mis-recognition made by our sequential solution and to take advantage of our knowledge of the expected answer, we propose to use the results of the GED computation to develop a backtracking process. By selecting strokes with an incorrect fit with the expected answer, we can call back the recognition step to

produce new segmentation hypotheses on these specific strokes in order to look for a better fit to the expected answer.

The rest of the paper is structured as follows. We present our graph representation and construction with an overview of our system in Section 2.1 before introducing the sub-segmentation of graphs in Section 2.2. We present the backtracking solution to correct potential recognition mistakes in Section 2.3. The experimental results on analysis accuracy and the quantity of operations timing-out on an in-house Handwritten Arithmetical Operation (HAO) dataset are detailed in Section 3. We discuss future improvements in Section 4.

2 Proposed method

2.1 Overview

In the following sections we present our contributions. We first present an overview of the system before presenting the graph of lines segmentation and backtracking steps.

The Figure 2 presents an overview of our system. (1) A graph is first constructed given a list of strokes as input : several classifiers are used namely for the strokes segmentation into symbols, the symbols classification and the symbols relationships classification. Using the classification rates of the previous classifiers, we will be able to use previous computation on pair of strokes or symbols to create new hypotheses. The segmentation features and classifier are based on [16]. The symbol classifier uses a standard VGG [17] architecture. The relationship features and classifier, computed using learned fuzzy landscapes [18] for mathematical relationship, are based on our previous work [19]. (2) Once the graph containing the symbols and their mathematical relationships is built, it is once again segmented using those same relationships to create a new graph of lines (see Section 2.2). This line segmentation is also applied to the expected answer graph.

(3) We can compute the GED using the DFS algorithm [9] to produce a first correspondence between our graphs of lines. Each corresponding pair of lines, represented each by a small subset of symbols, can later be matched to obtain a symbol-to-symbol correspondence. (5) If a pair of lines has a perfect match, it is possible to fix each corresponding pair of symbols, reducing the search space of the GED applied on the graph of symbols. (4) If any dissimilarities are detected between lines, we can backtrack to create new recognition hypotheses. By looking for new recognition hypotheses on non-corresponding strokes, we may correct mis-recognition by finding a better fit to the expected answer (see Section 2.3).

2.2 Graph of lines segmentation

In [4] the authors have shown that computing the exact GED on graphs larger than 16 vertices in less than 100 seconds is still challenging. To propose a system able to produce immediate feedback for the students, our goal is to reduce

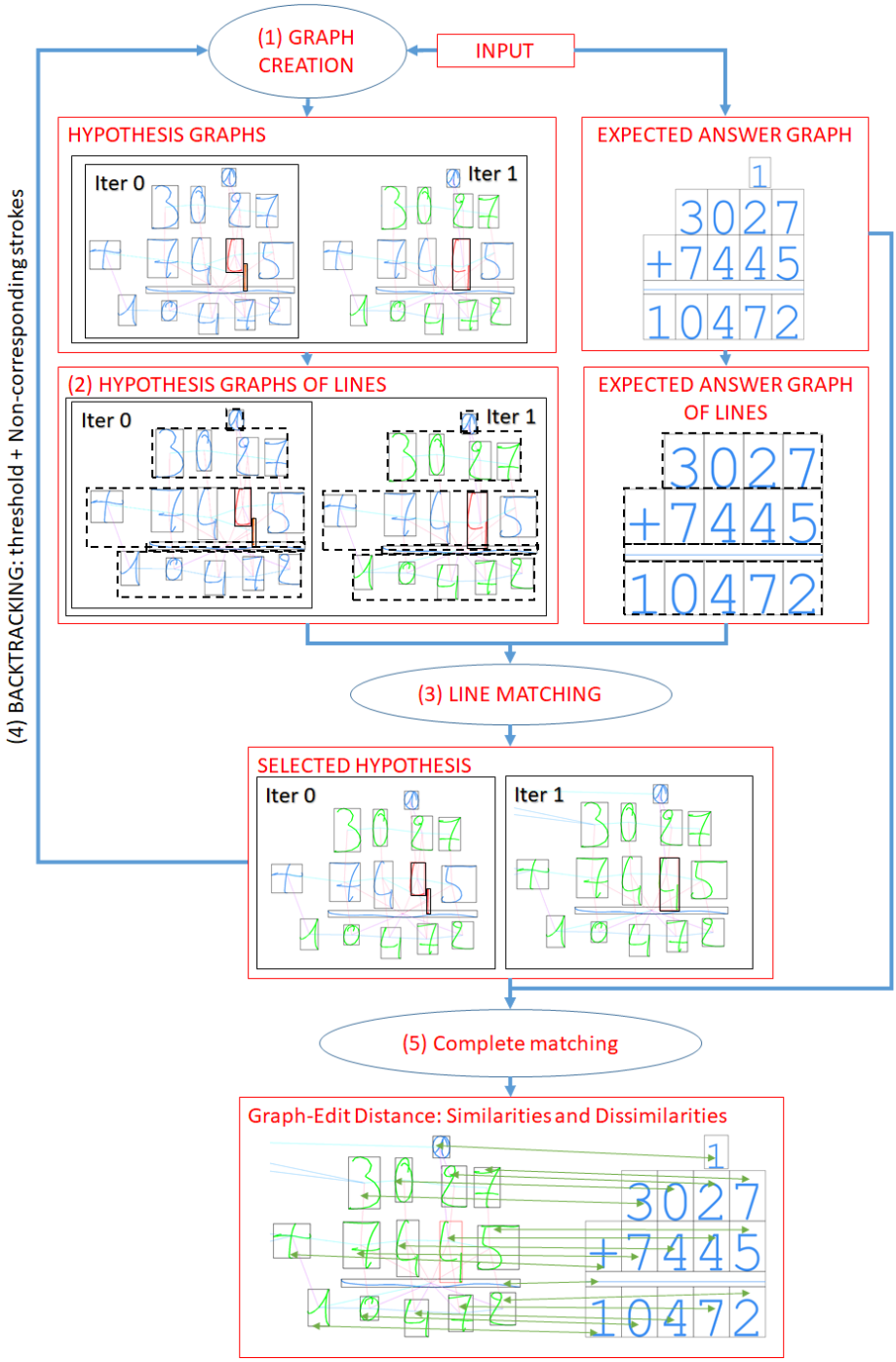

Fig. 2: Overview of our system. (1) From the input, a hypothesis graph and the expected answer graph are produced. (2) Those graphs are segmented into graphs of lines, (3) and then matched with the DFS algorithm. (4) If dissimilarities arise, the process is iterated with a lower threshold for non-corresponding strokes to create new hypotheses. (5) Once the steps are completed, the best hypothesis is kept and matched on the symbols level. In this example, the first iteration produce an incorrect segmentation on the highlighted digit 4. The corresponding line doesn't have a perfect match, thus we backtrack to the recognition process with a lower threshold. The second iteration create a new segmentation hypothesis on this digit, with a better fit to the expected answer (no dissimilarities arise).

this time-out to a couple of seconds at most. To overcome this computational problem, we propose to create a graph representation with a reduced number of vertices each containing more knowledge while keeping the structural identity of the operation.

An arithmetical operation is composed of numbers defined by the instruction, the expected result and single digits identified either as carry over or report. Those digits can have a variety of placement and be misaligned, however numbers are always written and aligned from left to right and each symbol constituting the number are temporally close one to another. As specified earlier, each operation is represented by a graph in which a vertex is a mathematical symbol and each edge represents a mathematical relationship between a pair of symbols. Using the relationships defined earlier, we can cluster those vertices into a sub-set of vertices, each one representing a line of the operation.

Algorithm 1 Graph of lines creation from a set of symbols and corresponding relationships

INPUT: V (set of symbols), E (set of relationships), t (temporal gap defined to reject symbols)
$V_{lines} = \{\}, E_{lines} = \{\}$
// We check every vertex in the graph
while V is not empty **do**
  // The nearest neighbor with Left or Right relationship are used to cluster the symbol vertex into a line vertex
  $search \leftarrow (V[0], Right), (V[0], Left)$
  $line \leftarrow \{V[0]\}$
  // Until no new neighbor with the corresponding relationship are found, we extend the neighbor search to new vertices
  while $search! = \phi$ **do**
    $v, r \leftarrow search[0]$
    for $v', r' \in E[v]$ **do**
      if $r' == r \&\& gap(v, v') < t$ **then**
        $search+ = (v', r')$
        $line+ = v'$
      end if
    end for
    $pop(search[0])$, $pop(V[v])$
  end while
  $V_{lines}+ = line$
end while
Each vertex is segmented into a line, and relationships between lines are deduced from relationships between vertex inside the lines
return V_{lines}, $inherit\_E(V_{lines}, V, E)$

Figure 3 shows the resulting Algorithm 1 applied on an addition. (a) A vertex is randomly selected, and linked to its neighbor with *Right* and $Left$ relationships. Those vertices are then selected and the search is extended on both sides with the corresponding edges. (b) Once no more vertices are found, a single vertex, containing the previously selected vertices, can be created. (c) The process is repeated until all vertices are checked to produce the complete graph of lines. To avoid clustering instruction vertices with their related carry over or report, a time constraint is considered. Given a vertex and a neighbor candidate with the correct relationship, if the temporal gap between them is too high, they are not considered belonging to the same cluster and thus the neighbor is rejected. The edges between clustered vertices one to another are inherited from the known relationships between symbols vertices. An expected answer graph of lines can also be generated with the same rules. This new representation contains less vertices, each vertex contains more information and the structural information between them is inherited from the graph of symbols representation.

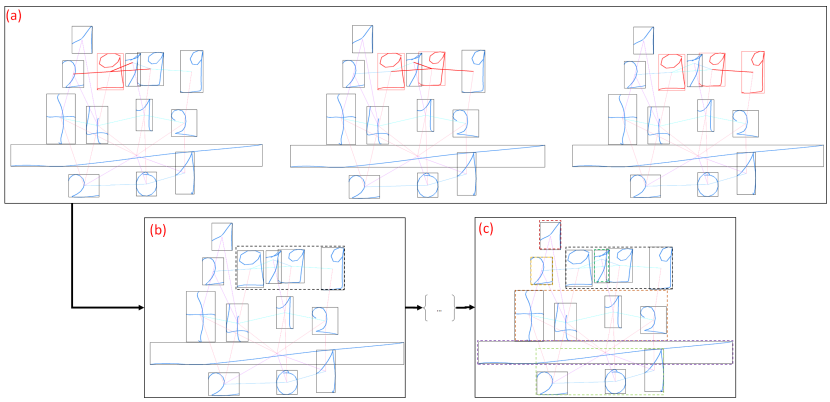

Fig. 3: An addition with 14 vertices written with the corresponding long arithmetic problem : $999 + 412 = 1411$. (a) A random vertex is selected and linked to its neighbor with Right and Left relationships. The highlighted edges between the 9 digits and the 1 and 2 digit are ignored because of the huge temporal gap between those vertices (carry over). The process is repeated for each neighbor until no new neighbor are found to create the line (b). The process is iterated until all vertices belongs to a line. (c) The resulting graph of lines contains 7 vertices.

Given this graph of lines representation, we can apply the DFS on this new pair of graphs. To compute the cost of replacing one line by another, we transform those sub-graphs into strings by using the left-right order inside each sub-graph. We can apply a Levenshtein distance to compute this cost. If two lines have a cost similarity of 0, we can fix them by adding this knowledge to the graph of symbols. To avoid an imperfect match between numbers due to the malleability

of the operators position, the insertion cost of an operator is fixed to 0. If at least a vertex has an incorrect label or is missing, other vertices belonging to the same cluster won't be directly matched.

The complete DFS on the pair of graph of symbols can be computed with the knowledge of fixed vertices with an updated cost (Equation 1). If at least one of the vertex was found and the id does not match, then an infinite cost is applied. Thanks to the updated cost for matched vertices, a large part of the search tree will be ignored. The DFS is computed much faster on initially larger graphs with the same resulting correspondence.

$$d(v_i, v_j) = \begin{cases} \text{if } found \text{ and } f_i! = f_j & \infty \\ \text{else if } \mu_i \in [\text{-}, +] \text{ and } \mu_j \notin [\text{-}, +] & 50 \\ \text{else if } \mu_i! = \mu_j & 10 \\ \text{else} & 0 \end{cases} \tag{1}$$

2.3 Backtracking recognition

Thanks to the line segmentation we are able to compute the DFS faster with the updated cost of corresponding symbols to non-corresponding symbols, thus considering only a sub-set of correspondence. However in addition to mistakes from the students, it is also possible to make wrong assumption on the recognition steps, either due to a wrong segmentation or by missing relationships with misaligned symbols. Thus vertices which should belong to the same cluster are split into different lines. In [19] we proposed to create multiple hypotheses using a fixed segmentation threshold before computing the matching. However on operations with many overlapping strokes, it is likely to create an exponential number of new hypotheses, many of whom will not fit the expected answer and could have been avoided. Moreover it is needed to compute a partial matching with each hypothesis to select the best fit, thus reaching a time-out if too many hypotheses are created.

We propose to include a backtracking step to take advantage of the resulting correspondence, detailed in the Algorithm 2. Figure 4 displays a backtracking example where a mis-segmentation on a digit can be revised by computing new segmentation hypotheses on the set of unmatched strokes. In the previous Section, we obtain a correspondence between graph of lines, which yield a cost transformation from lines to lines. If any lines from the expected answer is not perfectly matched with a transformation cost of 0, then non-corresponding strokes are labeled as such in the initial set of strokes. The recognition step is once again called with an increased threshold τ to create new hypotheses. Only the labeled pair of non-corresponding strokes are re-evaluated. If new hypotheses or relationships are created from one step to another, then the lines segmentation and matching is once again applied on each hypothesis. By reducing the search space to non-corresponding strokes, we can greatly reduce the number of new hypotheses that might be generated.

When multiple graphs are generated after a new recognition steps, each hypothesis is segmented into a graph of lines and matched to the expected answer.

The hypothesis with the lowest transformation cost is kept for further analysis while others are discarded. If the end condition is not reached (either a number of backtracking steps reached or all lines returning a perfect correspondence), then the recognition process is applied on non-corresponding strokes with an increased threshold τ. Previous segmentation hypotheses will be discarded, only new segmentation hypotheses as well as the current best will be kept for the new selection. Once the end condition is reached, we can compute the DFS on the remaining unmatched symbols on the best hypothesis selected.

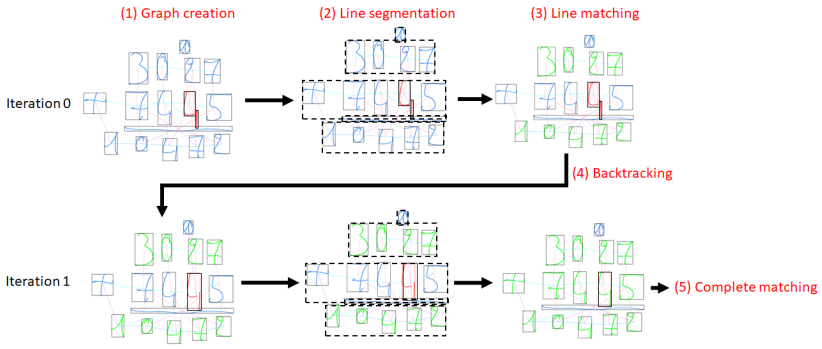

Fig. 4: Backtracking applied over an addition. The highlighted digit 4 is mis-segmented on the first iteration, resulting in an imperfect line matching. New segmentation hypotheses is computed on these un-matched strokes. The resulting new hypothesis has a better lines matching cost and new symbols can be preemptively matched. This hypothesis is kept for the DFS symbols-to-symbols.

3 Experiments

The dataset of handwritten arithmetical operation (HAO) presented in Table 1 is composed of both additions and substractions written by primary school students (age 6 to 9) on pen-based tablet. Samples from this dataset are displayed in Figure 5. An operation is considered incorrect when an expected symbol is missing (number, operator or carry over) or when an expected symbol is matched with an incorrect label. The result of an analysis is based on these mistakes. Symbols in excess are not counted as mistakes due to the noisy input devices but they are expected to not be matched to any symbol from the expected answer.

A separate training set of 200 operations was indiscriminately input by both adults and students and was used to train the segmentation and relation classifiers presented earlier. For the symbol classifier, a much larger training set from the CROHME 2019 competition [3] was used. The test set of 200 operations is

Algorithm 2 Recognition backtracking through lines matching

INPUT: S (set of strokes), τ (classification threshold), G_{EA} (expected answer graph), V_{fixed} (set of matched vertices)
// Generate a set of hypothesis given the threshold and previous found vertices
$\mathcal{G} \leftarrow reco(S, V_{fixed}, \tau)$
$G_{EA_{line}} \leftarrow line(G_{EA})$
cost = {}, path = {}
// For each hypothesis, apply the line segmentation and compute the correspondence with the expected answer
for $G \in \mathcal{G}$ **do**
  $G_{line} \leftarrow line(G)$
  $path, cost+ = DF - GED(G_{line}, G_{EA_{line}})$
end for
$path, cost = min(path, cost)$
// For each corresponding line in the best hypothesis, if its transformation cost is 0, keep the vertices
$V = V_{fixed}$
for $(line, line_{EA}) \in path$ **do**
  path_line, cost_line $\leftarrow$ Levenstein$(line, line_{EA})$
  if $cost\_line == 0$ **then**
    $V+ = line$
  end if
end for
// If all lines are matched, or if last iteration is reached, we return the correspondence symbol to symbol
if $(V == G_{EA_{line}} \| \tau == 0.2)$ **then**
  return DF-GED(best_hyp, G_{EA}, EP, V)
end if
// Otherwise, we backtrack with a greater threshold to generate more hypotheses while keeping the previous found vertices
return $backtrack(S, \tau + 0.05, G_{EA}, V)$

Table 1: Description of the HAO dataset. The size of the training and test set as well as the number of independent writers are reported. The number of mistakes contained through all operations and expected to be detected is also quantified. A single operation may contain multiple mistakes (see Figure 5).

	Training set	Test set
Size	200	200
Writers	28	24
Symbols	2908	3285
Incorrect symbols label	-	83
Missing symbols	-	122

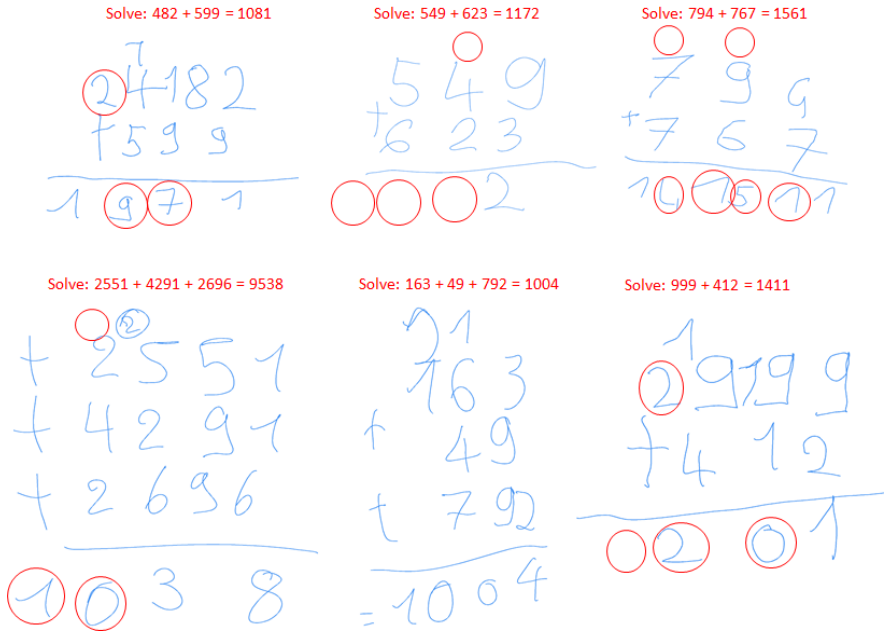

Fig. 5: Samples from the HAO dataset with their related long arithmetic problem. Students mistakes we are looking for are circled.

exclusively composed of handwritten operation written by primary school students. 110 out of 200 operations contains at least one mistake. No writer is found in both the training and testing set.

Evaluations are conducted on a machine with an Intel i5-8250U processor with 8GB of RAM. A time-out of 5 seconds was set for the computation on each operation. If the computation reaches a time-out, the last matching obtained is used for the analysis. We evaluate the improvement of the new graph segmentation in regards to the analysis score and quantity of operation reaching the time-out for different initial sizes of graphs. For the backtracking method, we consider a backtracking threshold value $\tau = 0.05\%$ with 5 backtracking iterations. The accuracy is computed on the detected dissimilarities between the operation and the expected answer compared to the ground truth (see Figure 8). If the dissimilarities detected are different, then the analysis result is considered incorrect.

We compare 4 different systems. (1) The DFS applied on the first graph of symbols hypothesis. (2) As proposed previously in [19], a partial matching applied on sub-graphs to match part of the operation before computing the DFS on the complete graph of symbols. (3) The DFS applied subsequently on the first graph of lines hypothesis then on the graphs of symbols with the fixed vertices. (4) The DFS applied in the complete backtracking process on several hypotheses and on the last selected graph of symbols hypothesis and its fixed vertices.

The goal is to evaluate the analysis score improvement of using the new graph of lines representation over different sizes of graphs in relation to the number of operations reaching a time-out. The results on the analysis accuracy are reported in the Figure 6. The analysis score are improved using the iterative process compared to previous methods. The backtracking allows for the correction of some mis-recognition from the system as we notice an improvement on the accuracy using the backtracking solution. The graph segmentation doesn't generate incorrect correspondence between graphs. We report the quantity of operations reaching a time-out in the Figure 7 for each method. We notice a much lower number of operations reaching the time out of 5 seconds by using the DFS on the graph of lines. The lines segmentation greatly improves the number of operations solved before reaching the time-out, which is in direct relation with operations in which an incorrect analysis was yielded before. The best process can match 162 out of 200 operations under 5 seconds, as opposed to the results previously reported in [19] of 134 operations.

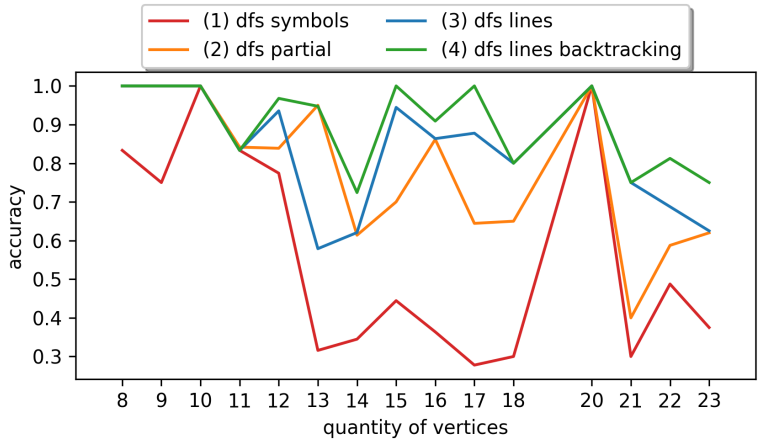

Fig. 6: Accuracy for each graph size. The accuracy is computed on the resulting analysis: if dissimilarities found were expected, the analysis is correct

Figure 8 displays two operations, the resulting recognition results from the backtracking process as well as the ground-truth which was previously annotated to evaluate the experiments. On the first operation, we are able to correct a mis-segmentation by keeping a new recognition hypothesis after a single iteration. However for the second operation, by searching a new recognition hypothesis with a better fit, we generated an hypothesis which yield an incorrect analysis. No hypothesis produces a correct matching on the result line because both segmentation do not correspond to the expected answer, and the system ends up selecting the incorrect segmentation hypothesis. Thanks to the lines matching, the DFS is computed before the time-out but an incorrect analysis is produced

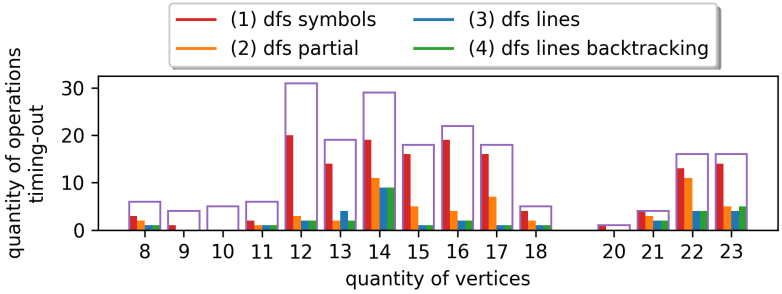

Fig. 7: Quantity of operations reaching time-out for each graph size confronted. The purple indicator represents the quantity of operations in the dataset for each graph size.

compared to the ground-truth. It would be necessary to have a balance between the fitting cost of the operation and the deterioration of the recognition results to avoid too far-fetched recognition hypothesis.

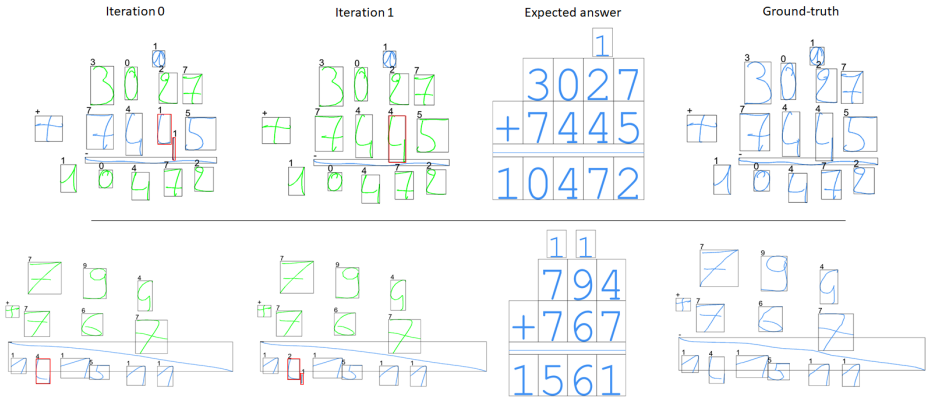

Fig. 8: Top: An operation with an initial incorrect segmentation, corrected thanks to the backtracking process on the second iteration. Bottom: incorrect segmentation on the highlighted 4 digit on the second iteration resulting in a similar high line matching cost, resulting in an incorrect hypothesis selected.

4 Conclusion and future works

We tackle the problematic of the analysis of on-line handwritten arithmetical operation in the context of primary school students long arithmetic teaching. We use a graph-based representation to transform the handwritten input. We

propose to use the knowledge of the expected answer and the correspondence between this answer and the student answer using Graph-Edit distance to correct recognition mistakes. To speed up the matching process and reduce the size of the search space, we propose a sub-segmentation to transform graphs of symbols to graphs of lines using the relative positioning and temporality between symbols. The structural identify of the operation is preserved while greatly reducing the graphs sizes and each vertex also contains more information. This allows for a faster yet correct matching between graphs. We introduce a backtracking step to produce several hypotheses for non-corresponding strokes. By doing so we produce new recognition hypotheses to look for a better fit to the expected answer.

The experiments on a dataset of 400 arithmetical operations show that we're able to improve the analysis results by correcting recognition mistakes through the backtracking steps. Those steps can be quickly computed thanks to the lines segmentation. The number of new hypothesis created doesn't increase exponentially by limiting the new recognition hypotheses to non-corresponding strokes. However relying on the correspondence with the expected answer might influence the system to select an incorrect recognition hypothesis with a better fit to the expected answer. To avoid those mistakes, it would be advised in future works to combine both the matching costs of hypotheses as well as the classifier scores for each hypothesis to avoid selecting a hypothesis degrading too much the recognition results.

References

1. Y. P. Xin, R. Tzur, C. Hord, J. Liu, J. Y. Park, and L. Si, "An intelligent tutor-assisted mathematics intervention program for students with learning difficulties," *Learning Disability Quarterly*, vol. 40, no. 1, pp. 4–16, 2017.
2. X. Huang, S. D. Craig, J. Xie, A. Graesser, and X. Hu, "Intelligent tutoring systems work as a math gap reducer in 6th grade after-school program," vol. 47, pp. 258–265, Elsevier, 2016.
3. M. Mahdavi, R. Zanibbi, H. Mouchere, C. Viard-Gaudin, and U. Garain, "Icdar 2019 crohme + tfd: Competition on recognition of handwritten mathematical expressions and typeset formula detection," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1533–1538, 2019.
4. D. B. Blumenthal and J. Gamper, "On the exact computation of the graph edit distance," *Pattern Recognition Letters*, vol. 134, pp. 46 – 57, 2020. Applications of Graph-based Techniques to Pattern Recognition.
5. J. Zhang, J. Du, and L. Dai, "Track, attend, and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition," *IEEE Transactions on Multimedia*, vol. 21, no. 1, pp. 221–233, 2018.
6. L. Hu and R. Zanibbi, "Mst-based visual parsing of online handwritten mathematical expressions," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 337–342, IEEE, 2016.
7. T. Zhang, H. Mouchère, and C. Viard-Gaudin, "A tree-blstm-based recognition system for online handwritten mathematical expressions," *Neural Computing and Applications*, pp. 1–20, 2018.

8. A. Sanfeliu and K.-S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE transactions on systems, man, and cybernetics*, no. 3, pp. 353–362, 1983.
9. Z. Abu-Aisheh, R. Raveaux, J.-Y. Ramel, and P. Martineau, "An Exact Graph Edit Distance Algorithm for Solving Pattern Recognition Problems," in *4th International Conference on Pattern Recognition Applications and Methods 2015*, (Lisbon, Portugal), Jan. 2015.
10. D. B. Blumenthal and J. Gamper, "Exact computation of graph edit distance for uniform and non-uniform metric edit costs," in *International Workshop on Graph-Based Representations in Pattern Recognition*, pp. 211–221, Springer, 2017.
11. J. Lerouge, Z. Abu-Aisheh, R. Raveaux, P. Héroux, and S. Adam, "New binary linear programming formulation to compute the graph edit distance," *Pattern Recognition*, vol. 72, pp. 254–265, 2017.
12. A. Fischer, R. Plamondon, Y. Savaria, K. Riesen, and H. Bunke, "A hausdorff heuristic for efficient computation of graph edit distance," in *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 83–92, Springer, 2014.
13. Y. Bai, H. Ding, S. Bian, T. Chen, Y. Sun, and W. Wang, "Simgnn: A neural network approach to fast graph similarity computation," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 384–392, 2019.
14. M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.
15. A. Dutta, P. Riba, J. Lladós, and A. Fornés, "Hierarchical stochastic graphlet embedding for graph-based pattern recognition," *Neural Computing and Applications*, vol. 32, no. 15, pp. 11579–11596, 2020.
16. L. Hu and R. Zanibbi, "Line-of-sight stroke graphs and parzen shape context features for handwritten math formula representation and symbol segmentation," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 180–186, IEEE, 2016.
17. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
18. A. Delaye and E. Anquetil, "Fuzzy relative positioning templates for symbol recognition," in *2011 12th International Conference on Document Analysis and Recognition (ICDAR)*, pp. 1220–1224, IEEE, 2011.
19. A. Lods, E. Anquetil, and S. Macé, "Graph edit distance for the analysis of children's on-line handwritten arithmetical operations," in *2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 337–342, IEEE, 2020.