Graph Edit Distance for the analysis of children's on-line handwritten arithmetical operations

Arnaud Lods, Eric Anquetil, Sébastien Macé

Graph Edit Distance for the analysis of children's on-line handwritten arithmetical operations

Arnaud Lods
Learn&Go, Univ Rennes, CNRS, IRISA
F-35000 Rennes
arnaud.lods@learn-and-go.com

Éric Anquetil
Univ Rennes, CNRS, IRISA
F-35000 Rennes
eric.anquetil@irisa.fr

Sébastien Macé
Learn&Go
F-35000 Rennes
sebastien.mace@learn-and-go.com

***Abstract*—This paper is based on a research project aiming at improving learning arithmetic operations at school using pen-based tablets. Given an arithmetic operation instruction, the goal is to analyze a child's handwritten answer. This comes down to find if any mistakes are made and their nature. An adapted representation and similarity search are needed for this analysis. In this paper, we propose to use a valued graph representation for handwritten arithmetical operations. To produce the analysis, we compute a similarity search with the corresponding expected answer using Graph Edit Distance (GED). To make up for the uncertainty of the noisy handwritten input recognition, we produce several segmented graph hypotheses for a single answer. Using the GED, we are able to correlate each hypothesis to the instruction graph. It enables to highlight multiple kinds of mistakes a child can make. The GED computation being a NP-complete problem, we propose to use sub-graph isomorphism: we partially match the instruction on each hypothesis in polynomial time to cut part of the tree search. Experiments were conducted on an in-house dataset composed of 400 handwritten arithmetical additions written by children on pen-based tablet. The time required for the GED computation is evaluated. We are able to match the complete operation in reasonable time on larger graphs while finding most of the time the best corresponding hypothesis.**

***Index Terms*—arithmetical operation analysis, graph matching, sub-graph isomorphism, graph edit distance**

I. Introduction

In the intelligent tutoring domain, several systems were proposed to provide an analysis of a student's answer for programming exercise [1] or for mathematics using keyboard interfaces [2], [3]. With the recent improvement of pen-tablet devices, such system can be enhanced to transfer solving mathematical problems from the devices to paper back and forth. The input process displayed in Figure 1 is straightforward: a child is given an operation instruction, and he is expected to solve it using a pen and a tablet as he would do it on paper. The resulting on-line handwritten input is a set of sequences of points in the 2D space. The answer expected from the child can be deduced from the instruction. The goal of the analysis system is to produce adapted feedback for the child. The feedback will depend on the nature of their mistakes. It is necessary to both recognize what was written and find any dissimilarities between the child's answer and the expected answer. Given the learning context, mistakes can be either misaligned symbols, a wrong instruction recopy, a forgetting of operators, calculus mistakes, unneeded symbols,

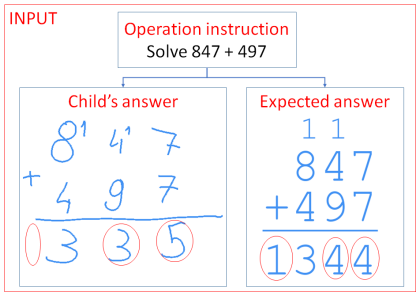

Fig. 1: Input of our system. An operation instruction is given to the child who wrote his answer on a pen-tablet device. An expected answer is generated based on the operation instruction. The dissimilarities we are looking for are circled.

an excess of carry over ... Other dissimilarities that will hinder the recognition are noisy input from the device and the malleability of handwriting.

We use a graph-based representation commonly used to represent mathematical expression [4]. By using such representation for both the input and the expected answer, it is possible to compute the Graph Edit Distance. It is a popular and general graph similarity computation [5] that searches the best vertices and edges correspondence between a pair of graphs. Two contributions are presented in this paper. We propose a way to compute and match several graph segmentation hypotheses to an instruction graph to select the most promising hypothesis. We propose a partial matching based on sub-graph isomorphism for our context of application to compute a better approximate matching and accelerate the complete Graph Edit Distance computation.

The paper is structured as follows. In Section II, we discuss the current state-of-the art on graph matching. We then define our graph representation and construction in Section III. In this, we present the transformation from raw input to a set of hypothesis graphs then we describe the process of matching multiple hypotheses with the expected answer graph. The results on an in-house Handwritten Arithmetical Operation (HAO) dataset are detailed in Section IV. We discuss future improvements in Section V.

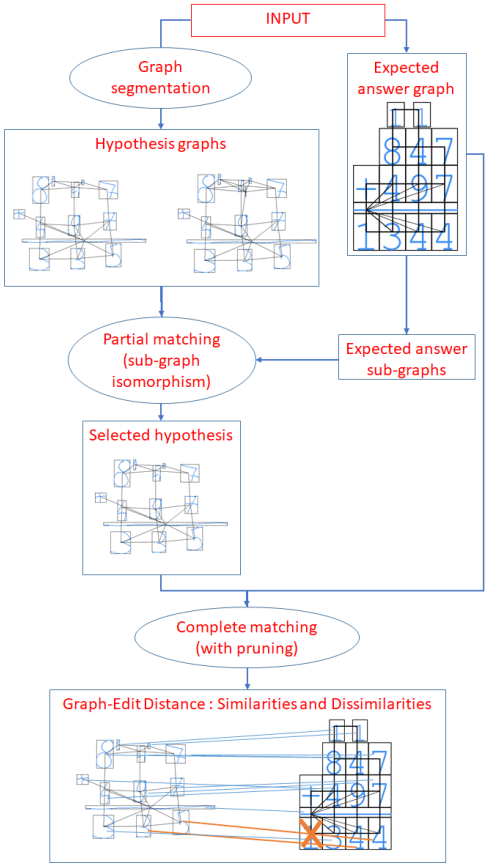

Fig. 2: Overview of our system. From the input, segmented hypothesis graphs are produced as well as an expected answer graph. The latter is matched to all hypothesis to find the best hypothesis to compute the GED. One result vertex is detected as missing and two result vertices have an incorrect label.

II. Problem Context

Handwritten mathematical expression (HME) recognition has been a widely researched subject [4] using end-to-end neural network or sequential solutions to transform the handwritten input into a valid mathematical expression. The recognized HME is usually represented by a graph with each vertex corresponding to a symbol and each edges representing a mathematical relationship. Figure 2 presents the workflow of our analysis system. The online handwritten strokes are transformed to a set of fuzzy visibility graphs [6] corresponding each to a different segmentation hypothesis. An adapted similarity search is necessary to select the most promising hypothesis and put into correspondence the expected answer. For this purpose we provide a quick lookup on methods tackling the task of Graph Edit Distance (GED) computation.

The popular GED computation used in the literature is mainly based on a tree search. A vertex in the tree corresponds to a partial edition of the graph and a leaf corresponds to a complete edit path transforming a graph G_1 into a graph G_2. The A* algorithm was the first used to complete this task but as a best-first search it is both computationally and memory expansive, and thus unusable on large graphs.

Systems for the inexact computation of the GED can often be computed faster by limiting the number of partial solutions [7], by using heuristic for efficient computation [8] or by using deep learning to learn an embedding function to map graph to vector for specific similarity measures [9]. They are efficient and useful for indexing or classification in large databases, but an approximation of the matching could result in an incorrect matching and provide wrong feedback to the child.

A comparison of several algorithms on the exact computation of the GED is proposed in [5]. The standard A* Best-First Search is evaluated as well as the Depth-First Search (DF-GED) algorithm, presented in [10]. They also compare an improvement using sub-graph isomorphism (CSI-GED) generalised for non-uniform edit costs [11]. Another compared method uses Binary Linear Programming [12] and the CPLEX mathematical solver. A lower bound can be computed either with Hungarian algorithm [13] or by solving a Quadratic Assignment Problem [14]. The paper comparing these algorithms concludes that it is now possible to compute the GED up to 16 vertices in reasonable time before frequently reaching time out. More recently a Beam-Stack Search was proposed in [15] but it only works for uniform edit costs which provide less expressiveness for graphs.

Given these methods, we propose to tackle our problem by using the DF-GED. The algorithm works for non-uniform edit costs which are adapted for arithmetical graphs as we can attribute more refined costs for edges with different mathematical relationships.

III. Analysis of Arithmetical Operation

In this section we present our contribution. We first specify the graph representation and segmentation. Then we present the proposed hypothesis selection through pre-matching and approximate matching before computing the GED on the most promising hypothesis.

A. Hypothesis graphs segmentation

Definition 1. *An attributed handwritten operation graph is a tuple G = (V, E, μ, ξ) where V is the set of vertices and E is the set of edges such that $\forall e = (i, j) \in E$, $i \in V$ and $j \in V$. $\mu : V \rightarrow L_v$ is the vertex labeling function that associates the label $\mu(v)$ to $v \in V$ and where $L_v = [0..9, +, -]$ is the set of possible labels for the vertices. $\xi : E \rightarrow L_e$ is the edge labeling function that associates a label $\xi(e)$ to $e \in E$ where $L_e = [Right, Left, Above, Below]$ is the set of labels for the edges.*

The graph is sequentially constructed using three different classifiers namely for the strokes segmentation into symbols, the symbols classification and the classification of relationships between symbols. We propose to generate several segmentation hypotheses to not induce early segmentation mistakes. A threshold τ is defined to induce several segmentation hypotheses using the probabilities output of the segmentation classifier. The segmentation classifier is similar to the one used in [16]; the Random Forest classifier output is a segmentation

rate for each pair of strokes. Using the threshold τ, we produce two different graphs if the segmentation probability is in between $[0.5-\tau; 0.5+\tau]$. The impact of τ on the segmentation will be evaluated in Section IV.

The relationship classifier presented in our previous work is also a Random Forest classifier using features computed from fuzzy landscapes [6]. For the symbol classification, we use a more general deep neural network inspired by VGG [17]. An iterative construction of the set of hypothesis graphs is straightforward and proposed in Algorithm 1.

Algorithm 1 Construction of the set of hypothesis graphs $\mathcal{G}$

INPUT: S (set of strokes)
V = {}, E = {}, $\mathcal{G}$ = {}
for all $s_1, s_2 \in S$ **do**
  merge = SEGMENTATION(s_1, s_2) [16]
  if $(0.5 - \tau \leq merge \leq 0.5 + \tau)$ **then**
    V += (s_1, s_2) + (s_1) + (s_2)
  else if $merge > 0.5 + \tau$ **then**
    V += (s_1, s_2)
  else
    V += (s_1) + (s_2)
  end if
  // create above a new vertex if stroke is not found, else merge with corresponding vertex
end for
for all $v \in V$ **do**
  $L_v = \mu(v)$
end for
for all $v_1, v_2 \in V$ **do**
  $E+ = \xi(v_1, v_2)$ [6]
end for
// for all unique set of vertices V' which covers all strokes, make a graph
for all unique set of vertices V' **do**
  $\mathcal{G}$ += (V', $E_{V'}$)
end for
return $\mathcal{G}$

B. Partial and complete matching

Given the operation instruction, we can generate an expected answer graph G_{EA}. This graph is computed using the algorithmic rules for each type of arithmetic operation. Given the set of hypothesis graphs $\mathcal{G}$ and the expected answer graph G_{EA}, we can compute the Graph Edit Distance to select the most promising hypothesis and find dissimilarities.

Definition 2. *The Graph Edit Distance (GED) is a measure of similarity between two graphs $G_1 = (V_1, E_1, \mu_1, \xi_1)$ and $G_2 = (V_2, E_2, \mu_2, \xi_2)$. An objective function is used to select the best edit-path, which is a set of operation o_i applied on G_1 to transform the graph into G_2. An operation o_i is either a vertex or edge substitution, deletion or insertion. The objective function to minimize f is: $f(G_1, G_2) = \min_{(o_1 \ldots o_k) \in T(G_1, G_2)} \sum_{i=1}^{k} c(o_i)$ where $T(G_1, G_2)$ is the set of all edit paths $o = (o_1, ..., o_k)$ that enables transforming G_1 to G_2. $c(.)$ is a cost associated to each edit operation o_i.*

Cost function and matrix Using fuzzy landscapes representing each mathematical relationship, we are able to evaluate relative positioning of pairs of symbols. We compute a membership to each fuzzy landscapes for a given edge $e_{(i,j)}$. The label L_e is a set of tuples (D, V_D) where $D \in [Right, Left, Above, Below]$ is an evaluated direction and V_D is its associated value. Given this relative positioning, we can compute the differences between a pair of edges with the equation:

$$R(e_{(i_1,j_1)}, e_{(i_2,j_2)}) = \sum_{d \in D} ||V1_d - V2_d|| * 100 \qquad (1)$$

where $e_{(i_1,j_1)}$ and $e_{(i_2,j_2)}$ are respectively edges from graphs G_1 and G_2.

Using our Eq. 1 to compute the differences between a pair of valued edges, it is possible to compute a single cost-matrix which takes into account both labels on vertices and their edges. This cost matrix dimensions are (n+m) X (m+n) where n is the number of vertices of a graph G_1 and m is the number of vertices of a graph G_2. The matrix is constructed as follows:

$$\left\| \begin{array}{ccc|ccc} c_{1,1} & .. & c_{1,m} & c_{1\to\epsilon} & \infty & \infty \\ .. & .. & .. & \infty & .. & \infty \\ c_{n,1} & .. & c_{n,m} & \infty & \infty & c_{n\to\epsilon} \\ \hline c_{1\leftarrow\epsilon} & \infty & \infty & 0 & 0 & 0 \\ \infty & .. & \infty & 0 & 0 & 0 \\ \infty & \infty & c_{m\leftarrow\epsilon} & 0 & 0 & 0 \end{array} \right\|$$

where $c_{i,j}$ is a substitution, $c_{i\leftarrow\epsilon}$ is a deletion and $c_{i\to\epsilon}$ is an insertion. The cost is computed so that both the vertices labels, the edges values and the adjacent vertices labels are taken into account to improve the accuracy of the lower-bound computation, with:

$$c_{v_i,v_j} = d(v_i, v_j) + \min_{t \in \mathcal{T}} \sum_{e_{(i,i')}, e_{(j,j')} \in t} R(e_{(i,i')}, e_{(j,j')}) + d(v_{i'}, v_{j'}) \qquad (2)$$

$$d(v_i, v_j) = \begin{cases} \text{if } \mu_i \in \text{[-, +]} \text{ and } \mu_j \notin \text{[-, +]} & 50 \\ \text{else if } \mu_i != \mu_j & 10 \\ \text{else} & 0 \end{cases} \qquad (3)$$

d is a function evaluating the cost transformation of a vertex label to another. $\mathcal{T}$ is the set of all combinations of edges from vertices v_i and v_j and t is the corresponding list of pair of edges $e(i, i'), e(j, j')$.

Depth First-GED (DF-GED) [10] is an efficient computation of the GED which uses upper and lower bounds with the Hungarian Algorithm and vertices sorting during the tree search. However it becomes computationally expensive on large graphs due to many backtracks in search for a better edit-path. Moreover with distant pairs of graphs such as bad segmentation hypotheses it is harder to prune large parts of

the tree search due to a lower bound too inaccurate. To face this, we propose to do a quick partial matching by searching a sub-graph isomorphism to match parts of the excepted graph answer. This way it provides a more accurate lower bound estimation for hypothesis selection and speeds-up the DF-GED search by applying it on a reduced tree search.

Definition 3. *Given the graph G_1 = (V, E, μ, ξ), a sub-graph G'_1 is a tuple G'_1 = (V'_1, E'_1, μ_1, ξ_1) where $V'_1 \subseteq V$ and $E'_1 \subseteq E$ where $\forall e'_{i,j} \in E'_1$, $i \in V'_1$ and $j \in V'_1$. The sub-graph G'_1 is isomorphic to a graph G_2 if there exists a sub-graph G'_2 such that $d(G'_1, G'_2) = 0$.*

In Figure 3 we display several ways to split the expected answer graph into smaller sub-graphs representing significant parts of the initial graph. The idea is to identify patterns in graphs that are more likely to be present in both graphs. Such pattern for arithmetical operations is the instruction recopy part; it is more likely to be found in the child's answer. Moreover, finding sub-graphs where most of the vertices have directed edges to other parts of the operation will make later pruning easier.

The task of searching for sub-graph isomorphism is NP-complete. However, by using the Hungarian algorithm, we can find the best approximate matching between a sub-graph G_{EA_i} and the graph G. By applying this algorithm, if the edit path between G_{EA_i} and a sub-graph of G yield an edit-cost of 0, then an isomorphism is found. This edit path is saved for the complete graphs. Then the DF-GED can be computed on the set of vertices between G_{EA} and G that are yet to be matched. Using this partial matching on each hypothesis graph, we can complete the partial edit-path using the Hungarian algorithm on each pair of graphs. The hypothesis with the lowest approximate matching cost is then selected to compute the DF-GED given the previous edit-path found.

Finding an isomorphism for large sub-graphs using the Hungarian algorithm output, even with refined cost, is unlikely. Instead it is possible to iterate this process over increasingly larger sub-graphs. A first sub-graph representing the instruction numbers (see Figure 3) is selected, then extended by adding the operators in a second step, and the carryovers in a third step. Two sub-graphs matching are evaluated in Section IV.

The complete matching is then processed using DF-GED given the partial matching found previously. This enables for both a direct tree search reduction on the number of vertices as well as a more precise lower-bound estimation thanks to edges directed on already matched vertices. The Algorithm 2 describes the process of the hypothesis selection before applying DF-GED.

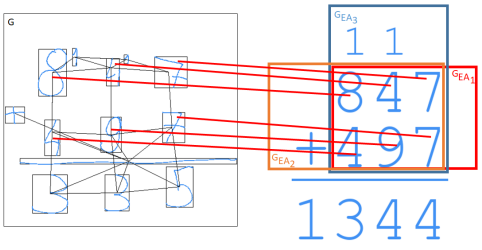

Fig. 3: Example of different sub-graphs for the expected answer graph. We can quickly find a perfect isomorphism for the small sub-graph G_{EA_1} with the graph hypothesis G. This sub-graph can be matched on the complete graph. It is possible iterate over larger sub-graphs to quickly find larger corresponding sub-graph isomorphism before computing the GED on the complete partially matched graph.

Algorithm 2 Hypothesis selection through partial matching

INPUT: $\mathcal{G}$ (set of hypothesis graphs), G_{EA} (expected answer graph), $\mathcal{G}'_{EA}$ (set of selected sub-graphs), EP (edit path, initially empty)
best_cost = inf, best_hyp = ϕ
for $G \in \mathcal{G}$ **do**
  best_path = {}
  for $G'_{EA_i} \in \mathcal{G}'_{EA}$ **do**
    // for each selected sub-graph, we look for a perfect match using the Hungarian algorithm
    Generate matrix C with best_path
    path = Hungarian(G, G_{EA_i}, C)
    if cost(G, G_{EA}, path) == 0 **then**
      best_path = path
    end if
  end for
  approx_cost = cost(G, G_{EA}, best_path) + Hungarian(G, G_{EA}, C)
  if (approx_cost<best_cost) **then**
    best_cost = approx_cost
    EP = best_path
    best_hyp = G
  end if
end for
// the DF-GED is computed on the remaining vertices given the edit path found for the best hypothesis
return DF-GED(best_hyp, G_{EA}, EP)

IV. Experimental Results

Our dataset of handwritten arithmetical operation (HAO) is composed exclusively of arithmetical additions. Samples from this dataset are displayed in Figure 4. An operation is considered incorrect when an expected symbol is missing (number, operator or carry over), when an expected symbol is matched with an incorrect label or when symbols are in excess. Strokes in excess due to the noisy input devices are not counted as mistakes.

The training set of 200 operations was indiscriminately input by both adults and children and was used to train the segmentation and relation [6] classifiers presented earlier. For the symbol classifier, a much larger training set from the CROHME 2019 competition [4] was used. The test set of 200 operations is exclusively composed of handwritten operation written by primary school children (age 7 to 9). 110 out of 200

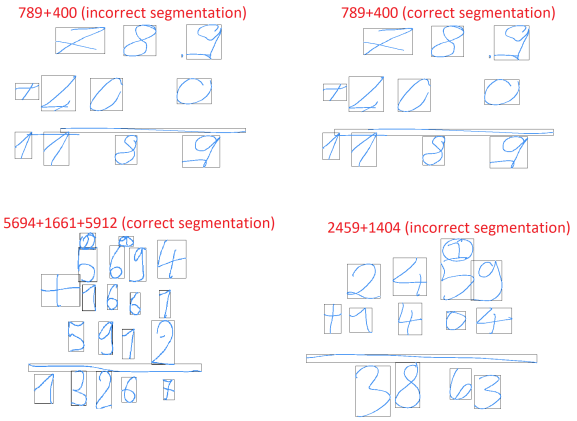

Fig. 4: Samples from the HAO dataset and their respective generated segmentation hypothesis.

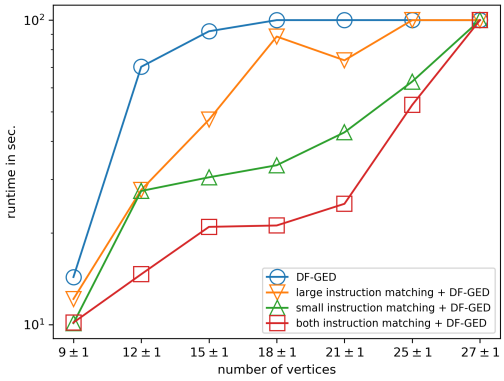

Fig. 5: Average time execution for different complete matching for increasing graphs length.

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. Running time are presented on log-scale and a time-out of 100 seconds was set for the computation on each operation. We have conducted two experiments. The first evaluates the contribution of the partial matching over the running time while the second evaluates both the impact of the segmentation threshold τ on the set of hypothesis graphs as well as the hypothesis selection running time.

We compare the running time of the standard DF-GED algorithm on the complete graphs to three different partial matching. We use the small sub-graph instruction (see Figure 3), a larger sub-graph with all vertices above the horizontal bar and then both sub-graphs. The DF-GED is then applied on the partially matched graphs. We evaluate the computation time for an increasing number of vertices. Figure 5 shows the consistent time improvement between the standard DF-GED and the reduced tree search of DF-GED thanks to the partial matching. In practice the DF-GED algorithm goes beyond the hour computation very fast, while the partial matching has still difficulties on larger graphs when no sub-graph isomorphism is found. In this case the system still reaches time-out on large arithmetical operations. On the set of 200 test operations, using only the DF-GED 20 operations are matched in less than 5 seconds (considered as reasonable for our use case) against 134 operations using both sub-graphs for partial matching. For the remaining operations, when no isomorphism is found, we could search for a partial matching with an edit-cost lesser than a fixed low threshold to avoid a time out on the complete GED.

For the second experiment we generate multiple segmentation hypotheses (Algorithm 1). To evaluate the impact of the segmentation threshold τ, we report in Table I for increasing values of τ the average and maximum number of hypothesis as well as the recall and precision. Our recall represents the presence of the ground-truth segmented hypothesis in the set of all generated hypotheses. Our precision corresponds to how often the selected hypothesis, using the best cost of approximate matching, was the ground-truth hypothesis.

The handwriting from children is complex and the same feedback could be produced from different segmented graphs. In the Figure 4, the top operation has a horizontal bar split into two strokes. Both strokes can be segmented either as a single symbol or as two separate symbols. In both cases, using the resulting edit-path we would find no mistakes in the operation (no symbols missing or wrong labels). However, to be more fair in our experiments, only one segmentation is considered correct for each operation (both strokes merged as a single horizontal bar in this example). As expected, with an increasing number of hypotheses, our recall is increased because the correct segmentation missed with $\tau = 0$ is generated with a larger segmentation threshold τ. The precision also increases as we are able to select the correct segmented hypothesis out of the set of generated hypotheses. However we are not able to reach a perfect segmentation and selection, as incorrect segmentation hypotheses might induce the same edit cost (see Figure 4).

TABLE I: Evolution of the number of hypotheses given an increasing segmentation threshold τ, and the corresponding recall and precision.

τ	# of hypotheses		Recall	Precision
	mean	max		
0	1	1	0.825	0.825
0.05	1.16	4	0.860	0.850
0.10	1.7	32	0.870	0.860
0.15	3.04	272	0.885	0.865
0.20	17.42	2880	0.935	0.900

Finally we evaluated the hypothesis selection over the increasing value of τ for different sizes of graphs. Figure 6 shows the computation time needed to achieve the hypothesis selection and the complete matching on the selected hypothesis. Using both partial and approximate matching we can quickly select the best hypothesis. With a larger value of τ the average running time is decreasing: when $\tau = 0$ a more distant

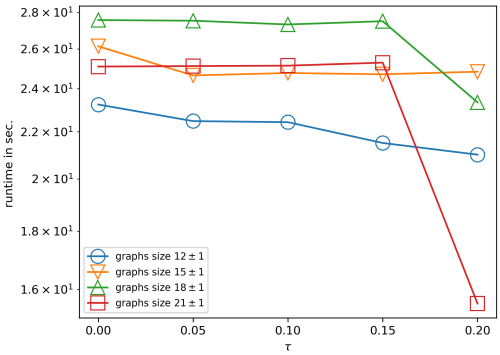

Fig. 6: Execution time for hypotheses selection and GED computation on the selected hypothesis for different sizes of graphs with a variable segmentation threshold τ.

hypothesis might be selected and induce a costlier DF-GED. With multiple hypotheses a better suited graph is selected with a better partial matching and a cheaper computation for the DF-GED on the remaining graph. However, for operations with a lot of overlapping strokes, too many hypotheses might be generated, in which case the system will run out of time. The solution to avoid an exponential number of hypotheses would be to have a dynamic τ depending on the number of generated hypotheses.

V. Conclusion and Future Works

We tackle the problematic of the analysis of on-line handwritten arithmetical operations in the context of children mathematics teaching. We propose the use of an adapted valued graph representation and a corresponding expected answer graph transformation to put into correspondence both graphs. We produce several graphs hypotheses to cover up for the uncertainty of the recognition results. We use subgraph isomorphism to partially match as much as possible the instruction to reduce the complexity of the Graph Edit Distance computation. This partial matching enables us to both select the best hypothesis with the lowest approximated matching cost and prune large parts of the depth-first search tree for the complete matching.

The experiments on an in-house dataset of 400 arithmetical additions show that from multiple hypotheses, the best one is selected most of the time in reasonable time. We are able to compute the matching on large graphs and much faster. However, the increasing number of hypotheses is a problem that could be solved either by using a better segmentation classifier or by adapting the segmentation threshold at the cost of a lower recall. Given the nature of our process, an incremental partial matching stroke by stroke while the child is writing could reduce the number of promising segmentation hypotheses. This would enable us to compute the Graph Edit Distance for future experiments on larger operations such as multiplications and divisions.

Acknowledgment

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

References

[1] V. J. Marin, T. Pereira, S. Sridharan, and C. R. Rivero, "Automated personalized feedback in introductory java programming moocs," in *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pp. 1259–1270, IEEE, 2017.

[2] 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.

[3] 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," *Learning and Individual Differences*, vol. 47, pp. 258–265, 2016.

[4] 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.

[5] 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.

[6] A. Lods, Anquetil, and S. Macé, "Fuzzy visibility graph for structural analysis of online handwritten mathematical expressions," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pp. 641–646, 2019.

[7] M. Neuhaus, K. Riesen, and H. Bunke, "Fast suboptimal algorithms for the 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. 163–172, Springer, 2006.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] K. Riesen, S. Fankhauser, and H. Bunke, "Speeding up graph edit distance computation with a bipartite heuristic.," in *MLG*, pp. 21–24, 2007.

[14] S. Bougleux, L. Brun, V. Carletti, P. Foggia, B. Gaüzère, and M. Vento, "Graph edit distance as a quadratic assignment problem," *Pattern Recognition Letters*, vol. 87, pp. 38–46, 2017.

[15] X. Chen, H. Huo, J. Huan, and J. S. Vitter, "An efficient algorithm for graph edit distance computation," *Knowledge-Based Systems*, vol. 163, pp. 762–775, 2019.

[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.