A differentiable approximation for the Linear Sum Assignment Problem with Edition

Luc Brun, Benoît Gaüzère, Guillaume Renton, Sébastien Bougleux, Florian Yger

A differentiable approximation for the Linear Sum Assignment Problem with Edition

Luc Brun*, Benoit Gaüzère†, Guillaume Renton†, Sébastien Bougleux*, Florian Yger‡
*Normandie Univ, ENSICAEN, CNRS, UNICAEN, GREYC, 14000 Caen, France
†Normandie Univ, INSA Rouen Normandie, LITIS, Rouen, France
‡PSL-Université Paris-Dauphine, CNRS, LAMSADE, Paris, France

***Abstract*—Linear Sum Assignment Problem (LSAP) consists in mapping two sets of points of equal sizes according to a matrix encoding the cost of mapping each pair of points. The Linear Sum Assignment Problem with Edition (LSAPE) extends this problem by allowing the mapping of sets of different sizes and adding the possibility to reject some matchings. This problem is set up by a rectangular cost matrix whose last column and last line encode the costs of rejecting the match of an element of respectively the first and the second sets. LSAPE has been the workhorse of many fundamental graph problems such as graph edit distance, median graph computation or sub graph matching. LSAP may be solved using the Hungarian algorithm while an equivalent efficient discrete algorithm has been designed for LSAPE. However, while the Sinkhorn algorithm constitutes a continuous solver for LSAP, no such algorithm yet exists for LSAPE. This lack of solvers forbids the integration of LSAPE in Neural networks requiring continuous operations from the input to the final loss. This paper aims at providing such a solver, hence paving the way to an integration of LSAPE solvers in Neural Networks.**

I. Introduction

Comparing structured data owing to the similarity of their substructures is a challenging task with many applications. Graph matching is the epitome of this problem and has a wide range of applications accross domains from chemoinformatics to computer vision. However, most approaches to do so either have a combinatorial flavour and do not play well with modern deep learning or do not allow slight errors in the matchings.

We propose in this paper a continuous estimation of an ϵ-assignment (Definition 1). Roughly speaking, an ϵ-assignment between two sets V_1 and V_2 may be understood as a bijective mapping between a sub set of V_1 and a sub set of V_2. The remaining elements of V_1 (not included in this mapping) are mapped onto an ϵ pseudo element of V_2. We say that such elements are deleted. Conversely, the remaining elements of V_2 correspond to the image of the ϵ pseudo element of V_1 (Figure 1). We say that these elements are inserted.

Let us note that if V_1 and V_2 have the same size, the bijective mapping induced by an ϵ-assignment may involve all elements of V_1, each element being mapped onto a single element of V_2. In this sense, an ϵ -assignment is more general than a bijective mapping. Moreover, the main advantage of an ϵ-assignment is that it provides us the freedom to not map all elements. This last property allows us to reject some mappings if for example, these mappings are associated to a large cost.

An ϵ-assignment function may be associated to an ϵ-assignment matrix (Figure 1(b)) just like any bijective mapping is associated to a permutation matrix. Given two sets, V_1 and V_2 of respective sizes n and m, an ϵ-assignment matrix is encoded by a $(n+1) \times (m+1)$ matrix, where $n+1$ and $m+1$ play respectively the roles of the ϵ element of V_1 and the one of V_2. The last column of index $m+1$ of such a matrix encodes the deletions while the last line encodes the insertions. By construction, there is a single 1 in each of the first n rows and m columns, the remaining elements being set to 0.

Given V_1 and V_2, one can define a $(n+1) \times (m+1)$ cost matrix encoding the cost of the mapping of any element of V_1 onto an element of V_2 as well as the cost of deleting each element of V_1 and inserting each element of V_2. Finding an ϵ-assignment minimizing the sum of mappings, deletions and insertions costs is a direct extension of the Linear Sum Assignment Problem (LSAP) called the Linear Sum Assignment Problem with Edition [1] (LSAPE). Given an ϵ-assignment matrix X and a cost matrix C, this cost may be formulated as:

$$\min_X \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} c_{i,j} x_{i,j}$$

where X is taken over all ϵ-assignment matrices.

We defined in previous works [2], [1], an adaptation of the Hungarian algorithm [3] which allows to find an optimal solution to the above problem in $\mathcal{O}(\min(n,m)^2 \max(n,m))$. However, while providing an optimal solution, this algorithm does not readily allow the computation of the gradient of the associated operation. This last drawback, does not allow to easily insert such an algorithm into a deep learning pipeline. On the other hand, the Sinkhorn algorithm [4], is based

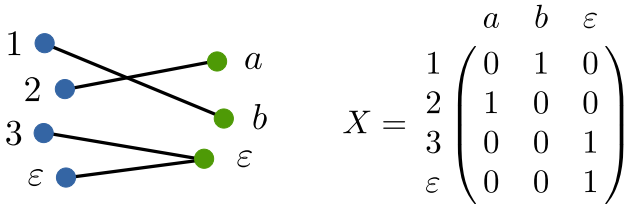

Fig. 1. Left: An example of ϵ-assignment function. 1 is mapped onto b, 2 onto a, 3 is deleted. Right: its associated ϵ assignment matrix

on a continuous relaxation of the problem where permutation matrices are replaced by bi-stochastic matrices with an entropic regularization. This algorithm is the workhorse of computational optimal transport [5] and is based on iterative matrix multiplications hereby allowing the backpropagation of the gradient [6]. The aim of this contribution is to transpose the results of the Sinkhorn algorithm to ϵ assignment matrices. Just like the Sinkhorn algorithm which does not provide a permutation matrix but rather a bi-stochastic matrix, our algorithm will provide an ϵ bi-stochastic matrix (Definition 2). This last point may be of advantage within the Neural Network framework where the hard decisions corresponding to ϵ-assignment matrices may not allow a proper propagation of the gradient.

More formally, given a similarity matrix S (which may be easily deduced from a cost matrix), we aim at finding two diagonal matrices D_1 and D_2 such that D_1SD_2 is an ϵ bi-stochastic matrix. Section II introduces the main concepts and provides two series of diagonal matrices converging respectively to two diagonal matrices D_1 and D_2 such that D_1SD_2 is ϵ bi-stochastic. Our algorithm provided in the same section is based on these series. Section III allows to convert, within the ϵ assignment framework, the problem of a maximization of sum into a problem of minimization. This last results allows us to compare in Section IV our results with optimal ones provided by discrete algorithms [2], [1] which perform a minimization.

II. A CONSTRUCTIVE ALGORITHM

Definition 1 (ϵ-assignment)**.**
Let n and m be two strictly positive integers. An ϵ-assignment is a mapping $\varphi : \{1,\dots,n+1\} \to \mathcal{P}(\{1,\dots,m+1\})$ satisfying the following constraints:

$$\begin{cases} \forall i \in \{1,\dots,n\}, & |\varphi(i)| = 1 \\ \forall j \in \{1,\dots,m\}, & |\varphi^{-1}(j)| = 1 \\ m+1 \in \varphi(n+1) \end{cases}$$

where $\mathcal{P}(\{1,\dots,m+1\})$ is the power set of $\{1,\dots,m+1\}$.

Each element of $i \in \{1,\dots,n\}$ is thus mapped onto a set composed of a single element of $\{1,\dots,m+1\}$ ($|\varphi(i)| = 1$) and in the same way the set of antecedents of each $j \in \{1,\dots,m\}$ is reduced to one element ($|\varphi^{-1}(j)| = 1$). Hence the only element of $\{1,\dots,n+1\}$ which can be mapped onto a set composed of several elements is $n+1$. In the same way, $m+1$ is the only element which may have several antecedents. The constraint $m+1 \in \varphi(n+1)$ ensures that $n+1$ is mapped to at least one element and that $m+1$ has at least an antecedent.

In the example of Figure 1 we have $n = 3$ and $m = 2$. Elements 1, 2, 3 are respectively mapped onto $\{b\}, \{a\}, \{\epsilon\}$. Where the last mapping corresponds to a deletion of 3 (which is mapped onto $m+1$). Consequently $m+1$ has two antecedents: 3 and $4 = n+1$.

Definition 2 (ϵ-bi-stochastic matrix)**.**
A non negative $(n+1) \times (m+1)$ matrix X is called an ϵ-bi-stochastic matrix iff:

$$\sum_{j=1}^{m+1} X_{i,j} = 1 \quad \forall i \in \{1,\dots,n\} \tag{1}$$

$$\sum_{i=1}^{n+1} X_{i,j} = 1 \quad \forall j \in \{1,\dots,m\} \tag{2}$$

$$x_{n+1,m+1} = 1$$

A matrix satisfying only equation (1) is called an ϵ-row stochastic matrix while a matrix satisfying only equation (2) is called an ϵ-column stochastic matrix. If $X \in \{0,1\}^{(n+1)\times(m+1)}$, X is called an ϵ-assignment matrix and there is a one-to-one mapping between ϵ-assignments and ϵ-assignment matrices [2], [1].

Let us note that any ϵ-bi-stochastic matrix is a bi-stochastic matrix on which the bi-stochastic constraints are relaxed on the last line and last column. So any squared bi-stochastic matrix is also an ϵ-bi-stochastic matrix (the reverse being obviously false).

A. Construction scheme

Let $A = (a_{i,j})$ denote a $(n+1) \times (m+1)$ matrix. For any $i \in \{1,\dots,n+1\}$ and any $j \in \{1,\dots,m+1\}$ let us consider the series $(x_{i,p})_{p\in\mathbb{N}}$ and $(y_{j,p})_{p\in\mathbb{N}}$ defined as follows:

$$\begin{cases} \forall i \in \{1,\dots,n\} & x_{i,p+1} = \chi_{i,p}^{-1} x_{i,p} \\ & x_{n+1,p} = 1 \\ \forall j \in \{1,\dots,m\} & y_{j,p+1} = \gamma_{j,p}^{-1} y_{j,p} \\ & y_{m+1,p} = 1 \end{cases}$$

with:

$$\begin{cases} \forall i \in \{1,\dots,n\} & x_{i,0} = \left(\sum_{j=1}^{m+1} a_{ij}\right)^{-1} \\ \forall j \in \{1,\dots,m\} & y_{j,0} = 1 \end{cases}$$

The two factors $\chi_{i,p}, i \in \{1,\dots,n+1\}$ and $\gamma_{j,p}, j \in \{1,\dots,m+1\}$ are defined as follows:

$$\begin{cases} \forall i \in \{1,\dots,n\} & \chi_{i,p} = \sum_{j=1}^{m+1} x_{i,p} a_{i,j} y_{j,p} \\ & \chi_{n+1,p} = 1 \\ \forall j \in \{1,\dots,m\} & \gamma_{j,p} = \sum_{i=1}^{n+1} \chi_{i,p}^{-1} x_{i,p} a_{i,j} y_{j,p} \\ & \gamma_{m+1,p} = 1 \end{cases}$$

Theorem II.1. ***A constructive method*** *Let A be a nonnegative $(n+1) \times (m+1)$ matrix with total support [7] such that $A[\{1,\dots,n\},\{1,\dots,m\}]$ does not contain any line or column filled with 0. Using the previously defined series, if all values of the last line and column of A are positive the series $(x_{i,p})_{p\in\mathbb{N}}, (y_{j,p})_{p\in\mathbb{N}}$, $i \in \{1,\dots,n+1\}, j \in \{1,\dots,m+1\}$ converge and define two series of diagonal matrices $D_{1,p} =$*

$diag(x_{1,p}, \ldots, x_{n+1,p})$ and $D_{2,p} = diag(y_{1,p}, \ldots, y_{m+1,p})$ which converge to a limit such that:

$$S = \lim_{p \to +\infty} D_{1,p} A D_{2,p}$$

is ϵ bi-stochastic.

Proof. The proof is detailed in [7]. Let us note that this proof uses different arguments than the one provided in [4] whose arguments do not hold for ϵ-assignments. □

The above theorem states both the existence of a rewriting of the matrix A into an ϵ bi-sotchastic matrix and a practical method to compute it. Let us note that under additional conditions, this decomposition is unique [7].

B. An iterative algorithm

Algorithm 1 Iterative algorithm

```
1:  function SINKHORN_D1D2(A ∈ R^{(n+1)×(m+1)}, nb_iter, eps)
2:      ones_n ← 1^{(n+1)}, ones_m ← 1^{(m+1)}
3:      y ← ones_m
4:      conv ← False, i ← 0
5:      while i ≤ nb_iter and not conv do
6:          xp ← 1/(A * y)
7:          xp[n] ← 1
8:          if i ≥ 1 then
9:              norm_x ← ‖ xp / x−ones_n‖
10:         end if
11:         x ← xp
12:         yp ← 1 /(A^⊤ * x)
13:         yp[m] ← 1
14:         norm_y ← ‖ yp / y−ones_m‖
15:         y ← yp
16:         if i ≥ 1 then
17:             conv ←norm_x < eps and norm_y < eps
18:         end if
19:         i ← i + 1
20:     end while
21:     return diag(x) * A * diag(y)
22: end function
```

The code (in python) corresponding to the construction of matrices $D_{1,p}$ and $D_{2,p}$ is provided in Alg. 1. You may note the fact that we set x_{n+1} and y_{m+1} to 1 respectively on line 7 and 13. This point together with the use of rectangular matrices is the main difference between this algorithm and the "classical" Sinkhorn algorithm. The convergence criterion which allows to avoid to loop up to the maximum number of iterations is based on the fact that both χ_p and γ_p converge toward a vector of 1. An equivalent code based on the computation of the matrix $S_p = D_{1,p} A D_{2,p}$ is provided in [7].

III. From similarity to cost matrices and vice versa

The Sinkhorn algorithm is well known for providing an approximation of the Linear Sum Assignment Problem (Section IV) which can be formulated as:

$$\max_X \sum_{i=1}^{n} \sum_{j=1}^{n} s_{i,j} x_{i,j}$$

where $S = (s_{i,j})$ is our similarity matrix and $X = (x_{i,j})$ is taken over all bi stochastic matrices. The optimal solution being a permutation matrix, hence a binary matrix.

This maximization problem may be translated into a minimization problem by considering the matrix $c\mathbb{1}_{n\times n} - S$, where $\mathbb{1}_{n\times n}$ is a $n \times n$ matrix filled of 1 and c is a positive constant greater than all values of S. We have indeed:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} (c - s_{i,j}) x_{i,j} = cn - \sum_{i=1}^{n} \sum_{j=1}^{n} s_{i,j} x_{i,j}$$

Hence c and n being constant, minimize $\sum_{i=1}^{n} \sum_{j=1}^{n} (c - s_{i,j}) x_{i,j}$ is equivalent to maximize $\sum_{i=1}^{n} \sum_{j=1}^{n} s_{i,j} x_{i,j}$. The matrix $c\mathbb{1}_{n\times n} - S$ is usually interpreted as a cost matrix. This last point is important if one wants to compare the Sinkhorn algorithm to an optimal Hungarian algorithm [3] which performs a minimization of costs instead of a maximization of similarities.

As stated in Section I, our algorithm being an extension of the Sinkhorn algorithm we expect it to converge to :

$$\max_X \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} s_{i,j} x_{i,j}$$

where $X = (x_{i,j})$ is taken over all ϵ bi-stochastic matrices. However, the transformation of this maximization of similarities into a minimization of costs, is slightly more complex in the case of ϵ assignment matrices. To do so, let us consider a positive constant c and a $(n+1) \times (m+1)$ matrix $C = (c_{i,j})$ with $c_{i,j} = 2c$ for $i \leq n$ and $j \leq n$ while $c_{i,m+1} = c$ and $c_{n+1,j} = c$ for $i \leq n$ and $j \leq m$ respectively. We further have $c_{n+1,m+1} = 0$. Considering the cost matrix $C - S$ we have:

$$\sum_{i=1}^{n+1} \sum_{j=1}^{m+1} (c_{i,j} - s_{i,j}) x_{i,j} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} c_{i,j} x_{i,j} - \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} s_{i,j} x_{i,j}$$

with:

$$\begin{aligned} \textstyle\sum_{i=1}^{n+1} \sum_{j=1}^{m+1} c_{i,j} x_{i,j} &= 2c \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} \\ &\quad + c \textstyle\sum_{i=1}^{n} x_{i,m+1} \\ &\quad + c \textstyle\sum_{j=1}^{m} x_{n+1,j} \\ &= c \textstyle\sum_{i=1}^{n} \sum_{j=1}^{m+1} x_{i,j} \\ &\quad + c \textstyle\sum_{j=1}^{m} \sum_{i=1}^{n+1} x_{i,j} \\ &= cn + cm \end{aligned}$$

We have thus:

$$\sum_{i=1}^{n+1} \sum_{j=1}^{m+1} (c_{i,j} - s_{i,j}) x_{i,j} = c(n+m) - \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} s_{i,j} x_{i,j} \quad (3)$$

The minimization of the left part of equation 3 (minimization of costs) is thus equivalent to a maximization of the similarities.

Let us note that the trivial solution consisting to take $C = c\mathbb{1}_{(n+1)\times(m+1)}$ does not provide an equivalence between both problems since additional terms related either to the last column or the last row forbid to state that one problem is equal to a constant minus the other problem.

IV. Experiments

We proposed in Section II an algorithm converging toward an ϵ bi-stochastic matrix if the conditions defined by Theorem II.1 are satisfied. The aim of this section is to measure experimentally the convergence of our algorithm toward a solution maximizing :

$$\sum_{i=1}^{n+1}\sum_{j=1}^{m+1} s_{i,j}x_{i,j}$$

over all ϵ bi-stochastic matrices X. Where $S = (s_{i,j})$ is the input matrix. Such a problem is called a Linear Sum Assignment Problem with Edition (LSAPE).

A. Deviation of the Sinkhorn algorithm from the optimal solution

Sinkhorn algorithm provides an approximate solution to the well known Linear Sum Assignment Problem (LSAP):

$$\max_X \sum_{i=1}^{n}\sum_{j=1}^{m} s_{i,j}x_{i,j}$$

where X is taken over the set of bi-stochastic matrices. From a certain point of view, LSAP may be considered as a restriction of the LSAPE with squared matrices and no deletions/insertions. Let us first evaluate the error induced by the use of the Sinkhorn algorithm. To do this, we define matrices filled by random number in the interval $[1, 2]$. For each matrix size, we generate 100 matrices and compute for each matrix both the solution produced by the Sinkhorn algorithm and the optimal one produced by an Hungarian algorithm. Relative errors of Sinkhorn algorithm given the ground truth computed by LSAP is approximately constant for all sizes of matrices and lies between 20% and 23%.

B. Deviation of our algorithms from the optimal solution

In order to test our algorithm we use the same kind of random matrices but with a specific procedure for the last row and column which encodes respectively the affinity of each element toward insertions and deletions:

$$S_{i,j} \coloneqq \begin{cases} \text{rand}()+1 & \text{if } i{\leq}n \wedge j{\leq}m \\ 0 & \text{if } i{=}n{+}1 \wedge j{=}m{+}1 \\ h\,\text{rand}() & \text{else} \end{cases} \tag{4}$$

For $h \leq 0.5$ we can insure that for any $(i, j) \in \{1, \ldots, n\} \times \{1, \ldots, m\}$, $s_{i,j} \geq s_{n+1,j} + s_{j,m+1}$. In other terms we always get a greater sum by substituting i onto j than by deleting i and then inserting j. Conversely, if $s_{i,j} < s_{n+1,j} + s_{j,m+1}$ the substitution of i onto j will never be part of an optimal ϵ-assignment since this operation can be replaced, with a greater value of the sum, by the removal of i and the insertion of j.

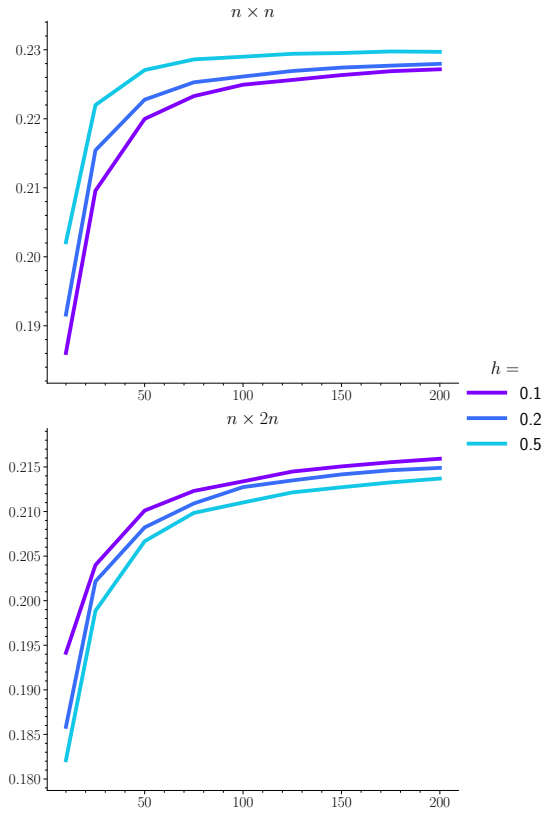

Fig. 2. Relative errors of our algorithm according to the optimal solution for $h \leq 0.5$.

Let us first focus on values of h lower than .5. Figure 2 shows the relative error of our algorithm according to an optimal LSAPE algorithm [2] for increasing sizes of the matrix. Let us note that [2] minimizes a sum of costs. We compare both algorithms using the results of Section III. For each matrix size, 100 random matrices are generated and the results are averaged for both algorithms (our algorithm and the optimal one). Our algorithm provides an approximation of the LSAPE which is slightly above 20%, hence comparable with the one provided by the Sinkhorn algorithm for the LSAP problem.

For h greater than .5 we observe in Figure 3 that we keep an error of about 20% for $h = 1.0$ and $h = 2.0$. In these cases the values of the last line and the last column remain comparable with the inner values of the random matrix. However, for larger values of h, namely $h \in \{4.0, 6.0, 8.0\}$ we observe a large increase of the relative error when the matrices are squared. We can conclude from these experiments that our algorithms do not converge to the expected value for squared matrices and when the values of the last column/line are very large compared to the inner values. More precisely, when we have:

$$s_{i,j} \ll s_{n+1,j} + s_{i,m+1} \text{ for } (i, j) \in \{1, \ldots, n\} \times \{1, \ldots, m\}$$

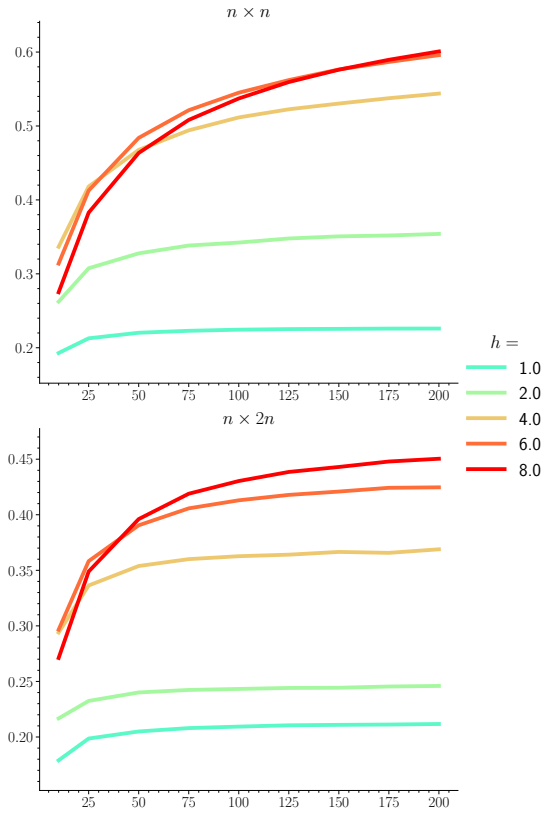

Fig. 3. Relative errors of our algorithm according to the optimal solution for $h > 0.5$.

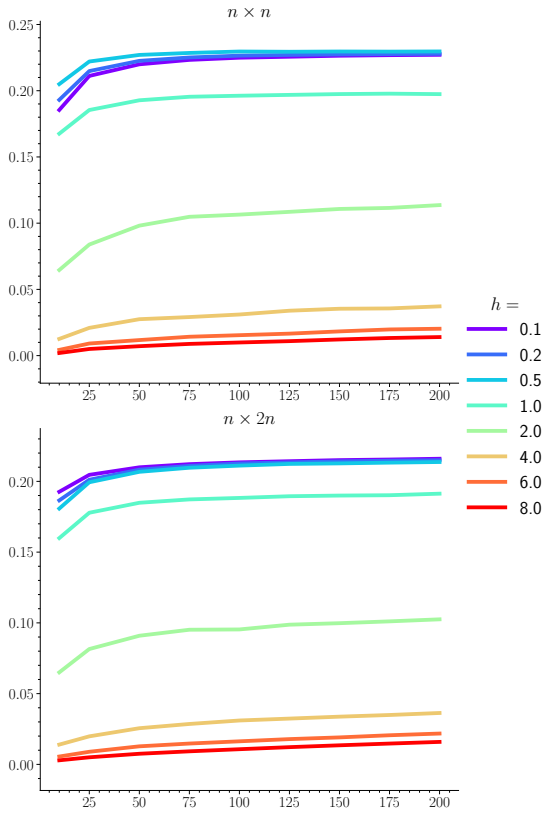

Fig. 4. Relative errors according to the optimal solution after a simplification of the similarity matrix.

A simple solution to fix this problem, consists in simplifying the similarity matrix by removing (setting to a low value) any entry (i, j), $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$ such that (i, j) cannot belong to any optimal solution. Given the similarity matrix S, such entries are characterized by $s_{i,j} < s_{n+1,j} + s_{i,m+1}$. In such cases, the substitution of i onto j may be advantageously replaced by the removal of i and the insertion of j. This last point forbids the assignment of i onto j in any optimal ϵ-assignment.

Figure 4 represents the relative errors according to the optimal solution performed by our algorithm using this simplification of the similarity matrix. In this experiment the "low value" replacing any entry of the matrix S which can not be included in any optimal solution has been fixed to 10^{-4}. One can first observe that we get the same behavior for the squared ($n \times n$) case and the rectangular one ($n \times 2n$). We can further observe that all errors remain below 20% for all sizes. Moreover, the relative error appear to be decreasing as a function of h in both cases. This may be explained by the fact that as h get higher, the simplified matrix becomes more and more trivial. Indeed for largest values of h, simplified similarity matrices correspond to trivial matrices with a constant value (equal to 10^{-4} in this experiment) for all entries (i, j) in $\{1, \ldots, n\} \times \{1, \ldots, m\}$ and a last row and column which remains unchanged and greater than 10^{-4} by several orders of magnitude. In such cases our algorithm converges to the optimal solution which corresponds to the removal of all elements in $\{1, \ldots, n\}$ and the insertion of all elements in $\{1, \ldots, m\}$.

C. Execution times of our algorithms

The execution times of our algorithm computed either on a graphic card (GeForce RTX 2080) or on a CPU (Intel(R) Xeon(R) Gold 5218 @2.30GHz) are displayed in Figures 5. Our algorithm has been run 100 times and the execution times have been averaged. In this experiment the range of values of n has been set to $[10, 2000]$.

Considering the squared case (first line of Figure 5) we observe an acceleration by a factor 2.2 thanks to the GPU. We can further observe on both figures a clear separation between curves corresponding to $h \leq 1$ and the ones corresponding to $h > 1$. As previously, this last point is due to the fact that as h get higher the simplified similarity matrices become more and more trivial and our iterative algorithms need less and less iterations to converge.

Concerning the rectangular case (second line, Figure 5), we observe a ratio equal to 9.4 between CPU and GPU execution times. The separation between both families of curves encountered in the square case is here less pronounced especially for GPU computations. Let us finally note, that the mean number of iterations required by our algorithm in the rectangular case is equal to 11.7 (mean computed over all

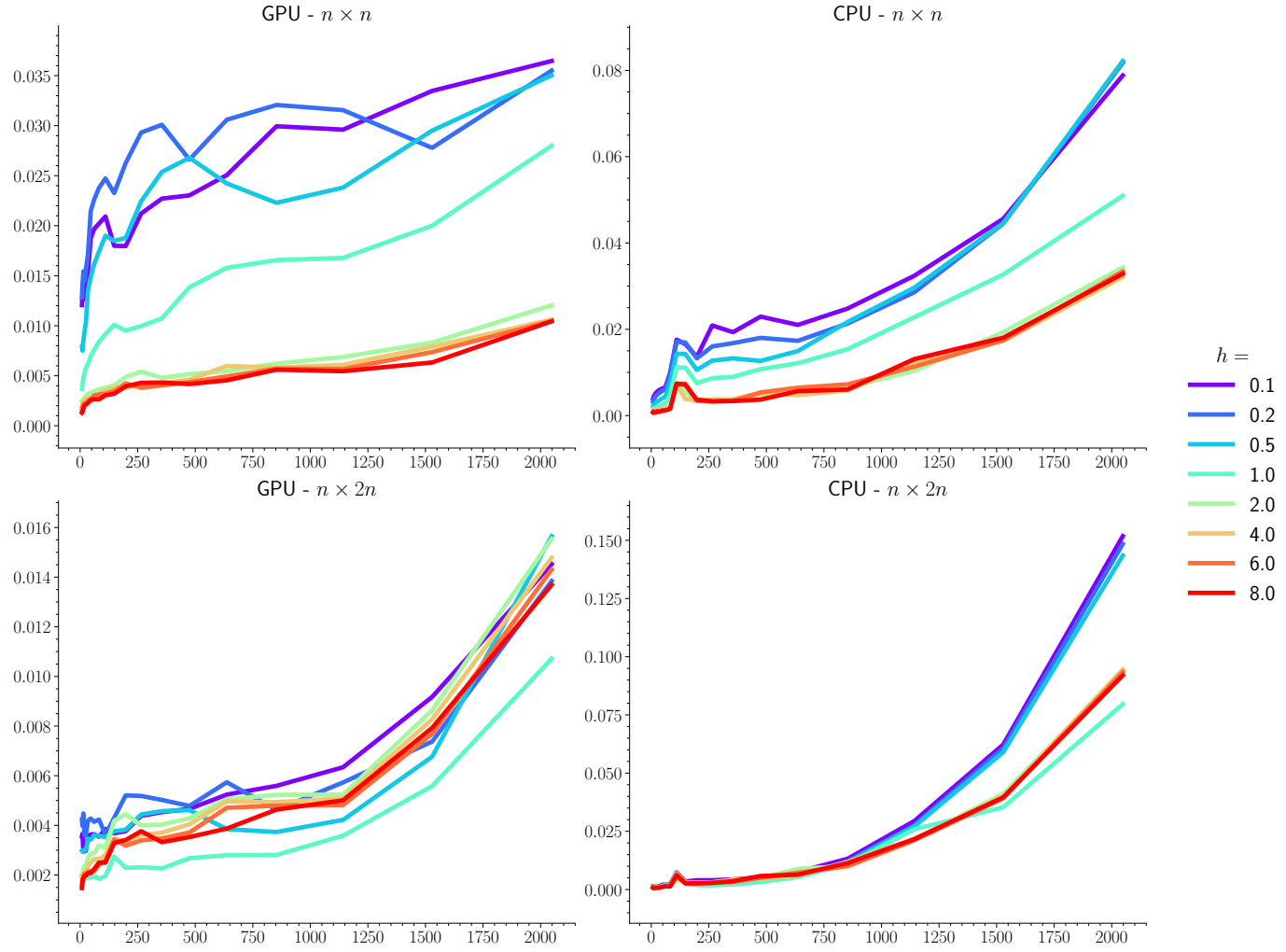

Fig. 5. Execution times using both graphic card and cpu computation using $n \times n$ matrices (first line) and $n \times 2n$ matrices (second line).

matrices' sizes and all values of h) while the one of the square case is equal to 45.8. This last point explains why, despite a data size that is doubled ($n \times n$ vs. $n \times 2n$), the execution times required by our algorithm in the rectangular cases are approximately 2 times less than those required in the square cases.

V. Conclusion

We have presented in this paper an algorithm providing an approximate solution to the Linear Sum Assignment Problem with Edition (LSAPE). The relative error of this algorithm compared to the optimal solutions is similar, and even much lower in some cases, to the relative error between the classical Sinkhorn and the optimal solutions to the Linear Sum Assignment Problem (LSAP). The main difference between this algorithm and the Hungarian based algorithms providing the optimal solution to the LSAPE is that our algorithm is iterative and differentiable and may thus be easily inserted within a backpropagation based learning framework such as artificial neural networks.

References

[1] S. Bougleux and L. Brun, "Linear Sum Assignment with Edition," Normandie Université ; GREYC CNRS UMR 6072, Research Report, Mar. 2016. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01288288

[2] S. Bougleux, B. Gaüzère, and L. Brun, "A Hungarian Algorithm for Error-Correcting Graph Matching," in *11th IAPR-TC-15 International Workshop on Graph-Based Representation in Pattern Recognition (GbRPR 2017)*, ser. Lecture notes in Computer Sciences (LNCS), P. Foggia, C.-L. Liu, and M. Vento, Eds., vol. 10310, Pasquale Foggia. AnaCapri, Italy: Springer, May 2017, pp. 118–127. [Online]. Available: PDF(HAL):=https://hal.archives-ouvertes.fr/hal-01540920/file/hungarian-algorithm-error.pdf, www page :=https://hal.archives-ouvertes.fr/hal-01540920

[3] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957. [Online]. Available: http://www.jstor.org/stable/2098689

[4] R. D. Sinkhorn and P. J. Knopp, "concerning nonnegative matrices and doubly stochastic matrices," *Pacific Journal of Mathematics*, vol. 21, no. 2, p. 343–348, 1967.

[5] G. Peyré, M. Cuturi *et al.*, "Computational optimal transport: With applications to data science," *Foundations and Trends® in Machine Learning*, vol. 11, no. 5-6, pp. 355–607, 2019.

[6] A. Genevay, G. Peyre, and M. Cuturi, "Learning generative models with sinkhorn divergences," in *Proceedings of the Twenty-First International*

Conference on Artificial Intelligence and Statistics, ser. Proceedings of Machine Learning Research, A. Storkey and F. Perez-Cruz, Eds., vol. 84. PMLR, 09–11 Apr 2018, pp. 1608–1617. [Online]. Available: https://proceedings.mlr.press/v84/genevay18a.html

[7] L. Brun, B. Gaüzère, S. Bougleux, and F. Yger, "A new sinkhorn algorithm with deletion and insertion operations," *CoRR*, vol. abs/2111.14565, 2021. [Online]. Available: https://arxiv.org/abs/2111.14565