



**HAL**  
open science

## Linear Sum Assignment with Edition

Sébastien Bougleux, Luc Brun

► **To cite this version:**

Sébastien Bougleux, Luc Brun. Linear Sum Assignment with Edition. [Research Report] Normandie Université, France; GREYC CNRS UMR 6072. 2016. hal-01288288v1

**HAL Id: hal-01288288**

**<https://hal.science/hal-01288288v1>**

Submitted on 14 Mar 2016 (v1), last revised 21 Mar 2016 (v3)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NoDerivatives 4.0 International License

# Linear Sum Assignment with Edition

Sébastien Bougleux and Luc Brun

Normandie Université  
GREYC UMR 6072  
CNRS - Université de Caen Normandie - ENSICAEN  
Caen, France

March 14, 2016

source code downloadable at <http://forge.greyc.fr/projects/lsape>

## Abstract

We consider the problem of transforming a set of elements into another by a sequence of elementary edit operations, namely substitutions, removals and insertions of elements. Each possible edit operation is penalized by a non-negative cost and the cost of a transformation is measured by summing the costs of its operations. A solution to this problem consists in defining a transformation having a minimal cost, among all possible transformations. To compute such a solution, the classical approach consists in representing removal and insertion operations by augmenting the two sets so that they get the same size. This allows to express the problem as a linear sum assignment problem (LSAP), which thus finds an optimal bijection (or permutation, perfect matching) between the two augmented sets. While the LSAP is known to be efficiently solvable in polynomial time complexity, for instance with the Hungarian algorithm, useless time and memory are spent to treat the elements which have been added to the initial sets. In this report, we show that the problem can be formalized as an extension of the LSAP which considers only one additional element in each set to represent removal and insertion operations. A solution to the problem is no longer represented as a bijection between the two augmented sets. We show that the considered problem is a binary linear program (BLP) very close to the LSAP. While it can be solved by any BLP solver, we propose an adaptation of the Hungarian algorithm which improves the time and memory complexities previously obtained by the approach based on the LSAP. The importance of the improvement increases as the size of the two sets and their absolute difference increase. Based on the analysis of the problem presented in this report, other classical algorithms can be adapted.

## 1 Introduction

Assigning the elements of some set to the elements of another, such that the resulting assignment satisfy some optimality conditions, is a fundamental combinatorial problem encountered in a wide variety of scientific fields [10, 4]. This report focusses on linear sum assignment problems, also known as weighted bipartite graph matching problems. The simplest example assumes that the two sets have the same cardinality.

**Problem 1** (LSAP). *Provided two finite sets  $\mathcal{U}$  and  $\mathcal{V}$  having the same cardinality  $n$ , assigning the  $n$  elements of  $\mathcal{U}$  to the  $n$  elements of  $\mathcal{V}$  can be described by a bijective mapping  $\mathcal{U} \rightarrow \mathcal{V}$ . The task of assigning an element of  $\mathcal{U}$  to an element of  $\mathcal{V}$  is penalized by a real non-negative cost function  $c: \mathcal{U} \times \mathcal{V} \rightarrow [0, +\infty)$ . The Linear Sum Assignment Problem (LSAP) consists in finding an optimal bijection*

$$\hat{\varphi} \in \operatorname{argmin}_{\varphi \in \mathcal{B}(\mathcal{U}, \mathcal{V})} \left\{ A(\varphi, c) \stackrel{\text{def.}}{=} \sum_{u \in \mathcal{U}} c(u, \varphi(u)) \right\}, \quad (1)$$

where  $\mathcal{B}(\mathcal{U}, \mathcal{V})$  is the set of bijections from  $\mathcal{U}$  to  $\mathcal{V}$ .

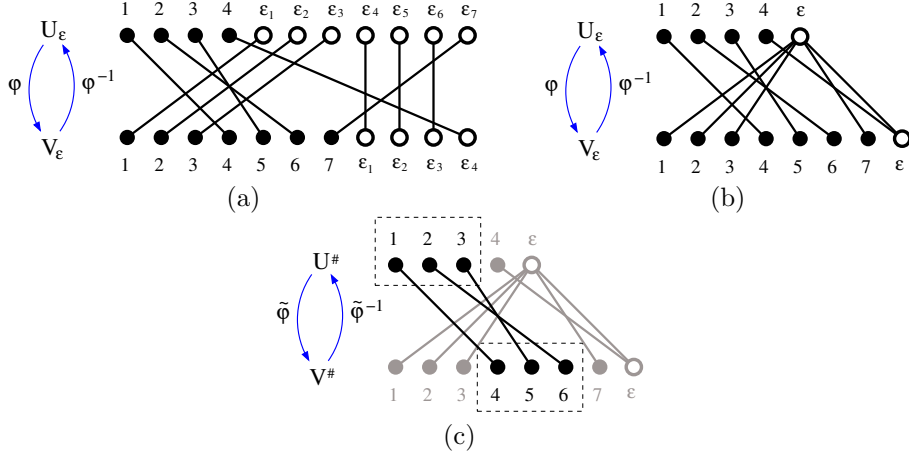


Figure 1: (a) A squared assignment with edition (Problem 2), *i.e.* a bijection  $\varphi: \mathcal{U} \cup \mathcal{E}_{\mathcal{U}} \rightarrow \mathcal{V} \cup \mathcal{E}_{\mathcal{V}}$ . (b) The equivalent assignment with edition (Problem 3), *i.e.* a mapping  $\varphi: \mathcal{U}_{\epsilon} \rightarrow \mathcal{P}(\mathcal{V}_{\epsilon})$  satisfying constraints given by Eq. 8. (c) The bijective restriction  $\tilde{\varphi}$  of the assignment with edition  $\varphi$  defined in (b). The sets  $\mathcal{U}^{\#}$  and  $\mathcal{V}^{\#}$  in (b) are surrounded by dashed boxes.

Note that several optimal assignments may exist, which depends on the cost function. Recall that any bijection is in one-to-one correspondence with a permutation of  $\{1, \dots, n\}$ , and that any permutation  $\varphi$  can be represented by a matrix  $\mathbf{X} \in \{0, 1\}^{n \times n}$  satisfying  $x_{i,j} = 1$  if  $\varphi(i) = j$  and  $x_{i,j} = 0$  else. Such a  $n \times n$  matrix, called permutation matrix, satisfies

$$\mathbf{X} \in \{0, 1\}^{n \times n}, \quad \mathbf{X}\mathbf{1}_n = \mathbf{1}_n, \quad \mathbf{X}^T\mathbf{1}_n = \mathbf{1}_n. \quad (2)$$

Let  $\mathcal{P}_n$  be the set of all  $n \times n$  permutation matrices. Then the LSAP can be reformulated as

$$\hat{\mathbf{X}} \in \operatorname{argmin}_{\mathbf{X} \in \mathcal{P}_n} \left\{ A(\mathbf{X}, \mathbf{C}) \stackrel{\text{def.}}{=} \sum_{i=1}^n \sum_{j=1}^n x_{i,j} c_{i,j} \right\}, \quad (3)$$

where  $\mathbf{C} = (c_{i,j})_{i,j=1,\dots,n}$  is the cost matrix representing the cost function, that is  $c_{i,j} = c(u_i, v_j)$  with  $u_i \in \mathcal{U}$  and  $v_j \in \mathcal{V}$ . From Eq. 3, the LSAP can be easily reformulated as a binary linear program [18]

$$\hat{\mathbf{x}} \in \operatorname{argmin} \left\{ \mathbf{c}^T \mathbf{x} \mid \mathbf{L}\mathbf{x} = \mathbf{1}_{n^2}, \mathbf{x} \in \{0, 1\}^{n^2} \right\}, \quad (4)$$

where  $\mathbf{x} = \operatorname{vec}(\mathbf{X})$  and  $\mathbf{c} = \operatorname{vec}(\mathbf{C})$  are the vectorization of the permutation matrix  $\mathbf{X}$  and the cost matrix  $\mathbf{C}$ , respectively. The right-hand side of Eq. 4 is the matrix version of the constraints defined by Eq. 2. The matrix  $\mathbf{L} \in \{0, 1\}^{2n \times n^2}$  corresponds to the node-edge incidence matrix of the complete bipartite graph  $K_{n,n}$  with node sets  $\mathcal{U}$  and  $\mathcal{V}$ , *i.e.*  $(\mathbf{L})_{k,(i,j)} = 1$  if  $(k = i) \vee (k = j)$  and 0 else. A solution to the LSAP corresponds to a perfect bipartite matching, *i.e.* a subgraph of  $K_{n,n}$  such that each element of  $\mathcal{U} \cup \mathcal{V}$  has degree one.

The LSAP is efficiently solvable. While there is  $n!$  possible assignments, it can be solved in polynomial time complexity (worst-case), for instance in  $O(n^3)$  with the well-known Hungarian algorithm [8, 9, 11] or its improvements, see [10, 4] for more details.

When the two sets  $\mathcal{U}$  and  $\mathcal{V}$  have different cardinalities,  $|\mathcal{U}| = n$  and  $|\mathcal{V}| = m$  with  $n \leq m$ , an assignment becomes an injection  $\varphi: \mathcal{U} \rightarrow \mathcal{V}$ , or equivalently an arrangement of  $n$  elements among  $m$ . There is  $n! C_m^n = m! / (m - n)!$  such assignments. Under this context, a solution to the LSAP is an injection minimizing the objective functional defined in Eq. 1. The Hungarian algorithm has been modified to solve this LSAP in  $O(n^2 m)$  time complexity [1].

In this paper, we study an extension of this problem which allows to model the transformation of the first set  $\mathcal{U}$  into the second one  $\mathcal{V}$  by means of edit operations. Three simple operations are considered. In addition to the traditional mapping of elements, *i.e.* each element of  $\mathcal{U}$  can be substituted by an element of  $\mathcal{V}$ , the elements of  $\mathcal{U}$  can be removed and the elements of  $\mathcal{V}$

can be inserted into  $\mathcal{U}$ . This extension is motivated by the definition of error-tolerant distance or similarity measures, which play an important role in pattern recognition. In particular, the graph edit distance [19, 3, 16, 2] is one of the most flexible graph dissimilarity measure, but computing it exactly is a NP-complete problem. The main strategy developed in the last decade consists in approximating the graph edit distance by solving a special weighted bipartite graph matching of the nodes such that insertion and removal operations are represented [7, 15, 12, 20, 14, 5, 6, 13, 17]. An edit path containing node and edge operations is then deduced from an initial node mapping provided by the LSAP. The approximate edit distance is then defined as the cost of this edit path. Initially designed as an upper bound of the graph edit distance by defining the cost of mapping two nodes as the cost of mapping their labels [7], it has been improved by also penalizing the mapping of their direct neighborhoods (incident edges and adjacent nodes) within this cost [15, 12]. More global and representative patterns than direct node neighborhoods, such as bags of walks [6], have also been explored.

The main difficulty within this bipartite framework consists in defining properly the removal and insertion operations. Moreover, the cost of mapping two direct neighborhoods or two bags is usually defined as the cost of an optimal weighted bipartite graph matching of the edges or patterns within the bags, which also consider removal and insertion operations. Such weighted bipartite graph matchings may be formulated as the following error-tolerant assignment problem between two sets [15, 12].

**Problem 2 (sLSAPE).** *Let  $\mathcal{U}$  and  $\mathcal{V}$  be two finite sets ( $|\mathcal{U}| = n$  and  $|\mathcal{V}| = m$ ). Any element of  $\mathcal{U}$  can be substituted to any element of  $\mathcal{V}$ . The removal of the elements of  $\mathcal{U}$  is represented by augmenting  $\mathcal{V}$  by  $n$  distinct null elements  $\mathcal{E}_{\mathcal{V}} = \{\epsilon_i\}_{i=1, \dots, n}$  such that each element  $\epsilon_i \in \mathcal{E}_{\mathcal{V}}$  corresponds to a unique element  $u_i \in \mathcal{U}$  and reciprocally. The insertion of the elements of  $\mathcal{V}$  in  $\mathcal{U}$  is represented similarly by augmenting  $\mathcal{U}$  by  $m$  elements  $\mathcal{E}_{\mathcal{U}} = \{\epsilon_j\}_{j=1, \dots, m}$ . The two augmented sets have the same cardinality  $n + m$  (Fig. 1(a)). Each possible edit operation (substitution, removal and insertion) is penalized by a cost function  $c: (\mathcal{U} \cup \mathcal{E}_{\mathcal{U}}) \times (\mathcal{V} \cup \mathcal{E}_{\mathcal{V}}) \rightarrow [0, \omega]$  such that:*

- the cost  $c(u_i, v_j)$  penalizes the substitution of any  $u_i \in \mathcal{U}$  by any  $v_j \in \mathcal{V}$ ,
- the cost  $c(u_i, \epsilon_i)$  penalizes the removal of any  $u_i$ ,
- the cost  $c(\epsilon_j, v_j)$  penalizes the insertion of any  $v_j$  into  $\mathcal{U}$ .

Assigning any  $u_i$  to any  $\epsilon_k$  ( $k \neq i$ ) or any  $\epsilon_l$  to any  $v_j$  ( $l \neq j$ ) is forbidden, so the associated costs are set to a large value  $\omega > \max\{c(u_i, v_j), \forall i \in \mathcal{U}, j \in \mathcal{V}\}$ , i.e.  $c(u_i, \epsilon_k) = c(\epsilon_l, v_j) = \omega$ . Finally, we have  $c(\epsilon_j, \epsilon_i) = 0$  for all  $u_i \in \mathcal{U}$  and all  $v_j \in \mathcal{V}$ , since assigning two null elements should not influence the overall assignment.

The squared linear sum assignment problem with edition (sLSAPE) consists in finding a bijection  $\varphi: \mathcal{U} \cup \mathcal{E}_{\mathcal{U}} \rightarrow \mathcal{V} \cup \mathcal{E}_{\mathcal{V}}$  that minimizes the objective functional:

$$\begin{aligned}
A(\varphi, c) &= \sum_{u \in \mathcal{U} \cup \mathcal{E}_{\mathcal{U}}} c(u, \varphi(u)) = \sum_{u_i \in \mathcal{U}} c(u_i, \varphi(u_i)) + \sum_{\epsilon_j \in \mathcal{E}_{\mathcal{U}}} c(\epsilon_j, \varphi(\epsilon_j)) \\
&= \underbrace{\sum_{\substack{u_i \in \mathcal{U} \\ v_j = \varphi(u_i)}} c(u_i, v_j)}_{\text{substitutions}} + \underbrace{\sum_{\substack{u_i \in \mathcal{U} \\ \epsilon_i = \varphi(u_i)}} c(u_i, \epsilon_i)}_{\text{removals}} + \underbrace{\sum_{\substack{\epsilon_j \in \mathcal{E}_{\mathcal{U}} \\ v_j = \varphi(\epsilon_j)}} c(\epsilon_j, v_j)}_{\text{insertions}}
\end{aligned} \tag{5}$$

The set of bijective mappings between  $\mathcal{U} \cup \mathcal{E}_{\mathcal{U}}$  and  $\mathcal{V} \cup \mathcal{E}_{\mathcal{V}}$  is denoted  $\mathcal{S}_{\mathcal{E}}(\mathcal{U}, \mathcal{V})$ .

Since the sLSAPE is a LSAP (Problem 1) with a specific cost function, it can be solved by any algorithm solving the LSAP, for instance in  $O((n + m)^3)$  time complexity by the Hungarian algorithm [12]. This algorithm is based on the formulation of the LSAP given by Eq. 3. For the sLSAPE, it consists in finding an optimal  $(n + m) \times (n + m)$  permutation matrix, provided

a  $(n+m) \times (n+m)$  edit cost matrix of the following form

$$\left( \begin{array}{cccc|cccc} & & & & c(1, \epsilon_1) & \omega & \cdots & \omega \\ & & & & \omega & c(2, \epsilon_2) & \ddots & \vdots \\ & & & & \vdots & \ddots & \ddots & \omega \\ & & & & \omega & \cdots & \omega & c(n, \epsilon_n) \\ \hline c(\epsilon_1, 1) & \omega & \cdots & \omega & & & & \\ \omega & c(\epsilon_2, 2) & \ddots & \vdots & & & & \\ \vdots & \ddots & \ddots & \omega & & & & \\ \omega & \cdots & \omega & c(\epsilon_m, m) & & & & \end{array} \right) \in [0, \omega]^{(n+m) \times (m+n)}. \quad (6)$$

As we can observe, many coefficients of such a cost matrix are not relevant (equal to  $\omega$  or to 0) in the blocks representing the cost of removal/insertion operations. These values have been added in order to model the problem of transforming one set into another, by means of edit operations, as a LSAP. To improve the time and memory complexity, the computation of an optimal transformation should not depend on these useless values. To this, a variant of the sLSAPE has been proposed in [17], but it restricts insertion (or removal) operations to occur in only one set.

In this report, rather than modelling the initial transformation problem such that it can be solved with known algorithms, we formalize it as an extension of the LSAP which considers only one additional element in each set to represent removal and insertion operations (Fig. 1(b)). A solution to the problem is not any more represented as a bijection between the two augmented sets (Section 2). We call this extension the linear sum assignment problem with edit operations (LSAPE). Intuitively, it consists in reducing the square edit cost matrix (Eq. 6) by removing its useless elements, in order to obtain a  $(n+1) \times (m+1)$  edit cost matrix

$$\left( \begin{array}{cccc|c} & & & & c(1, \epsilon) \\ & & & & c(2, \epsilon) \\ & & & & \vdots \\ & & & & c(n, \epsilon) \\ \hline c(\epsilon, 1) & c(\epsilon, 2) & \cdots & c(\epsilon, m) & 0 \end{array} \right) \in [0, +\infty)^{(n+1) \times (m+1)}. \quad (7)$$

We show that the LSAPE is a specific binary linear program where the set of constraints defines a mixed bipartite graph (Section 3). According to this analysis, we propose a modified Hungarian algorithm which solves the problem in  $O(\min\{n, m\}^2 \max\{n, m\})$  time complexity and in  $O(nm)$  memory complexity (Section 4). This is faster than the similar Hungarian algorithm used to solve the sLSAPE, as observed experimentally with comparable implementations.

## 2 Assignments with Edition

In this section we define the notion of assignments with edition (or  $\epsilon$ -assignments), and we analyze some of their algebraic and numerical properties.

### 2.1 Definition and combinatorial properties

Consider two finite sets  $\mathcal{U}$  and  $\mathcal{V}$ . Let  $n = |\mathcal{U}|$  and  $m = |\mathcal{V}|$  be their respective cardinalities. Transforming  $\mathcal{U}$  into  $\mathcal{V}$ , by means of edit operations, can be realized:

- by substituting each element of  $\mathcal{U}$  by a unique element of  $\mathcal{V}$ , or by removing it from  $\mathcal{U}$ , and then
- by inserting into  $\mathcal{U}$  each element of  $\mathcal{V}$  which has not been used for a substitution.

Let  $\epsilon$  be an element introduced to represent removal and insertion operations. The removal of an element  $u \in \mathcal{U}$  is denoted by  $u \rightarrow \epsilon$ , and the insertion of an element  $v \in \mathcal{V}$  into  $\mathcal{U}$  is denoted by  $\epsilon \rightarrow v$ . Consider the two augmented sets  $\mathcal{U}_\epsilon = \mathcal{U} \cup \{\epsilon\}$  and  $\mathcal{V}_\epsilon = \mathcal{V} \cup \{\epsilon\}$ . We propose to define a transformation from  $\mathcal{U}$  into  $\mathcal{V}$  as follows (Fig.1(b)).

**Definition 1** ( $\epsilon$ -assignment). An assignment with edition (or  $\epsilon$ -assignment) from  $\mathcal{U}$  to  $\mathcal{V}$  is a mapping  $\varphi: \mathcal{U}_\epsilon \rightarrow \mathcal{P}(\mathcal{V}_\epsilon)$  satisfying the following constraints:

$$\begin{cases} \forall u \in \mathcal{U}, & |\varphi(u)| = 1 \\ \forall v \in \mathcal{V}, & |\varphi^{-1}[v]| = 1 \\ \epsilon \in \varphi(\epsilon) \end{cases} \quad (8)$$

where  $\mathcal{P}(\cdot)$  is the powerset, and  $\varphi^{-1}[v] \subset \mathcal{U}_\epsilon$  denotes the preimage of any singleton  $\{v\} \in \mathcal{P}(\mathcal{V}_\epsilon)$  by  $\varphi$ . Note that braces are omitted for singletons ( $\varphi^{-1}[\{v\}]$  is written  $\varphi^{-1}[v]$ ). Let  $\mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  be the set of all  $\epsilon$ -assignments from  $\mathcal{U}$  to  $\mathcal{V}$ .

Note that the edit operation  $\epsilon \rightarrow \epsilon$  is always selected, but we could equivalently consider this operation as being never selected. Compared to the definition of bijections involved in the sLSPAЕ (Problem 2), an  $\epsilon$ -assignment does not constrain the  $\epsilon$ -elements directly. Indeed, by definition of  $\varphi$  above, we have

$$1 \leq |\varphi(\epsilon)| \leq m + 1 \quad \text{and} \quad 1 \leq |\varphi^{-1}[\epsilon]| \leq n + 1. \quad (9)$$

For example the  $\epsilon$ -assignment in Fig. 1(b) is given by

$$1 \rightarrow \{4\}, \quad 2 \rightarrow \{6\}, \quad 3 \rightarrow \{5\}, \quad 4 \rightarrow \{\epsilon\}, \quad \epsilon \rightarrow \{1, 2, 3, 7, \epsilon\}.$$

Definition 1 also implies the existence of the sets:

$$\begin{aligned} \mathcal{U}^\# &= \{u \in \mathcal{U} \mid \varphi(u) \subset \mathcal{V}\} = \varphi^{-1}[\mathcal{V}] \cap \mathcal{U} \\ \mathcal{V}^\# &= \{v \in \mathcal{V} \mid \varphi^{-1}[v] \subset \mathcal{U}\} = \varphi[\mathcal{U}] \cap \mathcal{V} \end{aligned} \quad (10)$$

where  $\varphi^{-1}[\mathcal{V}]$  denotes the preimage of the set composed of all singletons of  $\mathcal{P}(\mathcal{V})$  by  $\varphi$ . Similarly  $\varphi[\mathcal{U}]$  denotes the image of  $\mathcal{U}$  by  $\varphi$ . Then the function (Fig. 1(c))

$$\tilde{\varphi}: \begin{pmatrix} \mathcal{U}^\# & \rightarrow & \mathcal{V}^\# \\ u & \mapsto & v \text{ with } \{v\} = \varphi(u) \end{pmatrix} \quad (11)$$

is bijective. Indeed, given  $u \in \mathcal{U}^\#$ , by Eq. 8 we have  $|\varphi(u)| = 1$ . Since  $u \in \mathcal{U}^\#$ , there is  $v \in \mathcal{V}$  such that  $\varphi(u) = \{v\}$ . Moreover, by Eq. 8 we still have  $|\varphi^{-1}[v]| = 1$ , and since  $\varphi(u) = \{v\}$ , then  $\varphi^{-1}[v] = \{u\} \subset \mathcal{U}$ . Hence  $v \in \mathcal{V}^\#$ . Moreover, by Eq. 8, for any  $v \in \mathcal{V}^\#$  there is a single  $u \in \mathcal{U}$  such that  $\{v\} = \varphi(u)$ . We have  $u \in \mathcal{U}$  and  $\varphi(u) = \{v\} \subset \mathcal{V}$ , thus  $u \in \mathcal{U}^\#$ . Hence, for any  $v \in \mathcal{V}^\#$  there is a single  $u \in \mathcal{U}^\#$  such that  $\tilde{\varphi}(u) = v$ .

The function  $\tilde{\varphi}$  is thus bijective. It may be understood as the restriction of  $\varphi$  to the set of elements of  $\mathcal{U}$  which are not mapped onto  $\epsilon$ . An  $\epsilon$ -assignment may thus be understood as a bijection on which the bijectivity constraint is relaxed for  $\epsilon$ .

Let  $\mathcal{I}(\mathcal{U}, \mathcal{V})$  be the set of all injections from a subset of  $\mathcal{U}$  onto  $\mathcal{V}$ . Let  $\varphi \in \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  be an  $\epsilon$ -assignment. Note that the bijective restriction  $\tilde{\varphi}$  of  $\varphi$  is by definition injective from  $\mathcal{U}^\#$  onto  $\mathcal{V}$ , and hence belongs to  $\mathcal{I}(\mathcal{U}, \mathcal{V})$ . Consider a mapping  $\tilde{\varphi}: \mathcal{U}_s \subseteq \mathcal{U} \rightarrow \mathcal{V}$  of  $\mathcal{I}(\mathcal{U}, \mathcal{V})$ . Then the mapping  $\varphi: \mathcal{U}_\epsilon \rightarrow \mathcal{P}(\mathcal{V}_\epsilon)$  defined by:

$$\begin{cases} \forall u \in \mathcal{U}_s, & \varphi(u) = \{\tilde{\varphi}(u)\} \\ \forall u \in \mathcal{U} \setminus \mathcal{U}_s, & \varphi(u) = \{\epsilon\} \\ & \varphi(\epsilon) = \mathcal{V}_\epsilon \setminus \tilde{\varphi}[\mathcal{U}_s] \end{cases} \quad (12)$$

belongs to  $\mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  by construction. More generally  $\varphi$  and  $\tilde{\varphi}$  are linked as follows.

**Proposition 1.** The following mapping is bijective:

$$\psi: \begin{pmatrix} \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V}) & \rightarrow & \mathcal{I}(\mathcal{U}, \mathcal{V}) \\ \varphi & \mapsto & \tilde{\varphi} \end{pmatrix} \quad (13)$$

*Proof.* Consider a mapping  $\tilde{\varphi}: \mathcal{U}_s \subseteq \mathcal{U} \rightarrow \mathcal{V}$  of  $\mathcal{I}(\mathcal{U}, \mathcal{V})$ . The associated mapping  $\varphi \in \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  can be reconstructed according to Eq. 12, and one can easily show that  $\varphi$  belongs to  $\mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  by construction. The mapping  $\psi$  is thus surjective. We show that it is bijective. Consider two elements  $\varphi_1, \varphi_2 \in \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  such that  $\varphi_1 \neq \varphi_2$ . There is necessarily  $u \in \mathcal{U}_\epsilon$  such that  $\varphi_1(u) \neq \varphi_2(u)$ :

- If  $u \in \mathcal{U}$ , then:
  - If  $\varphi_1(u) = \{v\}$  and  $\varphi_2(u) = \{v'\}$  with  $\epsilon \notin \{v, v'\}$  then  $u \in \mathcal{U}_1^\# \cap \mathcal{U}_2^\#$ , but in this case we have  $\tilde{\varphi}_1(u) = v \neq \tilde{\varphi}_2(u) = v'$ , which implies that  $\tilde{\varphi}_1 \neq \tilde{\varphi}_2$ .
  - Else, suppose w.l.o.g. that  $\varphi_1(u) = \{\epsilon\}$  and  $\varphi_2(u) \neq \{\epsilon\}$ , then  $u \notin \mathcal{U}_1^\#$  and  $u \in \mathcal{U}_2^\#$  which implies that  $\tilde{\varphi}_1 \neq \tilde{\varphi}_2$  (the  $\tilde{\varphi}_i$  are not defined on the same subset of  $\mathcal{U}$ ).
- If  $u = \epsilon$ , consider w.l.o.g. that  $v \in \varphi_1(\epsilon) \setminus \varphi_2(\epsilon)$ . By Eq. 9, we have  $\epsilon \in \varphi_1(\epsilon) \cap \varphi_2(\epsilon)$ , consequently  $\epsilon \neq v \in \mathcal{V}$ . Since  $v \notin \varphi_2(\epsilon)$  it exists  $u' \in \mathcal{U}$  such that  $\varphi_2(u') = \{v\}$ . Then we have  $u' \in \mathcal{U}_2^\#$ .
  - If  $u' \notin \mathcal{U}_1^\#$ ,  $\tilde{\varphi}_1$  and  $\tilde{\varphi}_2$  are not defined on the same subset of  $\mathcal{U}$  and are consequently not equal.
  - If  $u' \in \mathcal{U}_1^\#$ , since  $v \in \varphi_1(\epsilon)$  then  $\varphi_1(u') \neq \{v\} = \varphi_2(u')$  from Eq. 8, and so  $\tilde{\varphi}_1 \neq \tilde{\varphi}_2$ .

In all cases we get  $\tilde{\varphi}_1 \neq \tilde{\varphi}_2$ . The mapping  $\psi$  is thus injective and hence bijective.  $\square$

A simple consequence is given by the following equality.

**Corollary 1.**  $|\mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})| = |\mathcal{I}(\mathcal{U}, \mathcal{V})|$ .

This allows to easily enumerate the number of  $\epsilon$ -assignments.

**Theorem 1.** *There is*

$$|\mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})| = \sum_{p=0}^{\min\{n,m\}} C_n^p C_m^p p!$$

possible  $\epsilon$ -assignments between two sets  $\mathcal{U}$  and  $\mathcal{V}$  with  $n = |\mathcal{U}|$ ,  $m = |\mathcal{V}|$ .  $C_n^p$  is the number of permutations (bijections), and  $p! C_m^p$  the number of partial permutations (injections).

*Proof.* Consider the set  $\mathcal{I}(\mathcal{U}, \mathcal{V})$  and the notations of Eq. 12. We have  $C_n^p$  subsets  $\mathcal{U}_s$  of  $\mathcal{U}$  with  $|\mathcal{U}_s| = p \leq \min\{n, m\}$ . For each subset there exists  $p! C_m^p$  injective mappings from  $\mathcal{U}_s$  to  $\mathcal{V}$ . So for each  $p = 0, \dots, \min\{n, m\}$ , the number of injective mappings is  $p! C_n^p C_m^p$ . Note that  $p = 0$  corresponds to the case where all elements of  $\mathcal{U}$  and  $\mathcal{V}$  are assigned to  $\epsilon$ .  $\square$

Remark that there exists much more  $\epsilon$ -assignments than injections. Indeed, suppose that  $n \leq m$ , we have

$$|\mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})| = n! C_m^n + \sum_{p=0}^{n-1} p! C_m^p C_n^p > n! C_m^n$$

However, there is much less  $\epsilon$ -assignments than bijections enumerated over  $n + m$  elements, as considered by the sLSAPE.

**Corollary 2.** *If w.l.o.g we suppose that  $n \leq m$  we have:*

$$C_{n+m}^n \leq |\mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})| \leq \frac{(n+m)!}{m!} < (n+m)!$$

*Proof.* We have by a classical result on binomial coefficient:

$$\sum_{p=0}^n C_n^p C_m^p = C_{n+m}^n = \frac{(n+m)!}{n!m!}$$

Hence from Theorem 1 we have:

$$|\mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})| \leq n! \sum_{p=0}^n C_n^p C_m^p = \frac{(n+m)!}{m!} < (n+m)!$$

$$|\mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})| \geq \sum_{p=0}^n C_n^p C_m^p = C_{n+m}^n$$

$\square$

As for bijections,  $\epsilon$ -assignments can be equivalently represented by matrices, which are useful for analyzing the associated linear sum assignment problem and for deriving a solution.

## 2.2 Matrix form

Given two sets  $\mathcal{U} = \{u_i\}_{i=1,\dots,n}$  and  $\mathcal{V} = \{v_j\}_{j=1,\dots,m}$ , any  $\epsilon$ -assignment  $\varphi \in \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  is in one-to-one correspondence with a matrix  $\mathbf{X} \in \{0, 1\}^{(n+1) \times (m+1)}$  such that its  $n$  first rows represent  $\mathcal{U}$  (row  $i$  corresponds to  $u_i$ ), its  $m$  first columns represent  $\mathcal{V}$  (column  $j$  corresponds to  $v_j$ ), its row  $(n+1)$  and its column  $(m+1)$  represent the null element  $\epsilon$ , and

$$\begin{cases} x_{i,j} = \delta_{\varphi(u_i)=\{v_j\}}, & \forall i=1, \dots, n, \forall j=1, \dots, m & \text{(substitutions)} \\ x_{i,m+1} = \delta_{\varphi(u_i)=\{\epsilon\}}, & \forall i=1, \dots, n & \text{(removals)} \\ x_{n+1,j} = \delta_{\varphi^{-1}[v_j]=\{\epsilon\}}, & \forall j=1, \dots, m & \text{(insertions)} \\ x_{n+1,m+1} = 1 & & \end{cases}$$

where  $\delta_r = 1$  if the relation  $r$  is true, or  $\delta_r = 0$  else.

The matrix  $\mathbf{X}$  has the general form:

$$\begin{array}{c} u_1 \\ \vdots \\ u_n \\ \epsilon \end{array} \left( \begin{array}{c|c} v_1 \cdots v_m & \epsilon \\ \hline \mathbf{X}^{\text{sub}} & \mathbf{x}^{\text{rem}} \\ \hline \mathbf{x}^{\text{ins}} & 1 \end{array} \right) \in \{0, 1\}^{(n+1) \times (m+1)} \quad (14)$$

where  $\mathbf{X}^{\text{sub}} \in \{0, 1\}^{n \times m}$  represents substitutions,  $\mathbf{x}^{\text{rem}} \in \{0, 1\}^{n \times 1}$  removals, and  $\mathbf{x}^{\text{ins}} \in \{0, 1\}^{1 \times m}$  insertions. For example, the matrix which represents the  $\epsilon$ -assignment shown in Fig. 1(b) is given by:

$$\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \epsilon \end{array} \left( \begin{array}{ccccccc|c} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & \epsilon \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{array} \right)$$

Due to the constraints on  $\varphi$  (Definition 1), such a matrix has a 1 on each of its  $n$  first rows, and a 1 on each of its  $m$  first columns:

$$\begin{cases} \sum_{j=1}^{m+1} x_{i,j} = 1, & \forall i=1, \dots, n & \text{(i.e. } |\varphi(u_i)| = 1) \\ \sum_{i=1}^{n+1} x_{i,j} = 1, & \forall j=1, \dots, m & \text{(i.e. } |\varphi^{-1}[v_j]| = 1) \\ x_{n+1,m+1} = 1 & & \end{cases} \quad (15)$$

Reciprocally, any matrix  $\mathbf{X} \in \{0, 1\}^{(n+1) \times (m+1)}$  satisfying Eq. 15 represents an  $\epsilon$ -assignment  $\varphi$  such that:

$$\begin{cases} \forall u_i \in \mathcal{U}, \quad \varphi(u_i) = \begin{cases} \{v_j\} & \text{if } x_{i,j} = 1 \text{ with } j \in \{1, \dots, m\} \\ \{\epsilon\} & \text{else (i.e. } x_{i,m+1} = 1) \end{cases} \\ \varphi(\epsilon) = \{v_j \in \mathcal{V}_\epsilon \mid x_{n+1,j} = 1\} \end{cases} \quad (16)$$

Let  $\mathcal{S}_{n,m,\epsilon}$  be the set of all matrices in  $\{0, 1\}^{(n+1) \times (m+1)}$  satisfying Eq. 15. It is the matrix form of  $\mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$ . Compared to a permutation matrix, where each row and each column sum to 1, the last row and the last column of a matrix satisfying Eq. 15 satisfies the following relaxed constraints (Eq. 9):

$$1 \leq \sum_{i=1}^{n+1} x_{i,m+1} \leq n+1 \quad \text{and} \quad 1 \leq \sum_{j=1}^{m+1} x_{n+1,j} \leq m+1.$$



Remark that any  $\epsilon$ -assignment  $\varphi \in \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  can be equivalently represented by a bipartite graph  $G = ((\mathcal{U}_\epsilon, \mathcal{V}_\epsilon), \mathcal{E})$  such that each edit operation performed according to  $\varphi$  corresponds to an edge of  $\mathcal{E}$ . This graph is a subgraph of the complete bipartite graph  $K_{n+1, m+1}$  generated from the two sets  $\mathcal{U}_\epsilon$  and  $\mathcal{V}_\epsilon$ . Note that the matrix  $\mathbf{X}$  associated with  $\varphi$  corresponds to the node adjacency matrix of  $G$ . Reciprocally, any subgraph of  $K_{n+1, m+1}$  represented by a node adjacency matrix  $\mathbf{X}$  satisfying Eq. 15 defines an  $\epsilon$ -assignment.

### 2.3 Partial assignments with edition

A partial assignment is an injection, *i.e.* an assignment wherein all elements are not assigned. It is in one-to-one correspondence with a partial permutation matrix. This can be defined similarly in the context of  $\epsilon$ -assignments.

**Definition 2** (partial  $\epsilon$ -assignment). *A partial  $\epsilon$ -assignment from  $\mathcal{U}$  to  $\mathcal{V}$  is a mapping  $\varphi : \mathcal{U}_\epsilon \rightarrow \mathcal{P}(\mathcal{V}_\epsilon)$  satisfying the following set of constraints:*

$$\begin{cases} \forall u \in \mathcal{U}, & |\varphi(u)| \in \{0, 1\} \\ \forall v \in \mathcal{V}, & |\varphi^{-1}[v]| \in \{0, 1\} \\ \epsilon \in \varphi(\epsilon) \end{cases} \quad (17)$$

which relaxes the one defining  $\epsilon$ -assignments (Eq. 8)

A partial  $\epsilon$ -assignment from  $\mathcal{U}$  to  $\mathcal{V}$  is an  $\epsilon$ -assignment from a subset  $\mathcal{U}' \subseteq \mathcal{U}$  onto a subset  $\mathcal{V}' \subseteq \mathcal{V}$ , *i.e.* an  $\epsilon$ -assignment in  $\mathcal{A}_\epsilon(\mathcal{U}', \mathcal{V}')$ .

A partial  $\epsilon$ -assignment can be equivalently represented by a matrix  $\mathbf{X} \in \{0, 1\}^{(n+1) \times (m+1)}$ , having the structure of matrices of  $S_{n, m, \epsilon}$  (Eq. 14), and satisfying the set of constraints

$$\begin{cases} \sum_{j=1}^{m+1} x_{i,j} \leq 1, & \forall i = 1, \dots, n \\ \sum_{i=1}^{n+1} x_{i,j} \leq 1, & \forall j = 1, \dots, m \\ x_{n+1, m+1} = 1 \end{cases} \quad (18)$$

This relaxes the ones defined by Eq. 15.

For example, the following matrix represents a partial  $\epsilon$ -assignment with two unassigned elements:  $u_2$  and  $v_2$ .

$$\begin{array}{c|cccccccc|c} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & \epsilon \\ \hline u_1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ u_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ u_3 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ u_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \epsilon & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{array}$$

Note that the matrix  $\mathbf{X}$  defines the node adjacency matrix of a bipartite graph having some isolated nodes.

## 3 Minimal Linear Sum Assignment with Edition

In order to simplify the forthcoming expressions, we assume w.l.o.g. that  $\mathcal{U} = \{1, \dots, n\}$ ,  $\mathcal{V} = \{1, \dots, m\}$ ,  $\mathcal{U}_\epsilon = \mathcal{U} \cup \{n+1\}$  and  $\mathcal{V}_\epsilon = \mathcal{V} \cup \{m+1\}$ , *i.e.* the element  $\epsilon$  corresponds to  $n+1$  in  $\mathcal{U}_\epsilon$  and to  $m+1$  in  $\mathcal{V}_\epsilon$ .

### 3.1 Edit cost and problem formulation

The definition of an  $\epsilon$ -assignment from  $\mathcal{U}$  to  $\mathcal{V}$  does not rely on the nature of the sets, *i.e.* on the nature of the underlying data. In order to select a relevant  $\epsilon$ -assignment, among all assignments from  $\mathcal{U}$  to  $\mathcal{V}$ , each possible edit operation  $i \rightarrow j$  is penalized by a non-negative cost  $c_{i,j}$ . All possible costs can be represented by an edit cost matrix  $\mathbf{C} \in [0, +\infty)^{(n+1) \times (m+1)}$  having the same structure as the one of matrices in  $S_{n,m,\epsilon}$  (Eq. 14):

$$\mathbf{C} = \left( \begin{array}{c|c} 1 \cdots m & \epsilon \\ \hline \mathbf{C}^{\text{sub}} & \mathbf{c}^{\text{rem}} \\ \hline \mathbf{c}^{\text{ins}} & 0 \end{array} \right) \begin{array}{c} 1 \\ \vdots \\ n \\ \epsilon \end{array} \quad (19)$$

where  $c_{i,j}^{\text{sub}}$  penalizes the substitution  $i \rightarrow j$  for all pair  $(i, j) \in \mathcal{U} \times \mathcal{V}$ ,  $c_{i,m+1}$  penalizes the removal  $i \rightarrow \epsilon$  for all  $i \in \mathcal{U}$ ,  $c_{n+1,j}$  penalizes the insertion  $\epsilon \rightarrow j$  for all  $j \in \mathcal{V}$ , and  $c_{n+1,m+1} = 0$  associates a zero cost to the mapping  $\epsilon \rightarrow \epsilon$ . Note that the edit cost matrix associates a cost to each edge of the complete bipartite graph  $K_{n+1,m+1}$ .

Let  $\varphi \in \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  be an  $\epsilon$ -assignment. Its cost is defined as

$$\begin{aligned} A_\epsilon(\varphi, \mathbf{C}) &= \sum_{i \in \mathcal{U}_\epsilon} \sum_{j \in \varphi(i)} c_{i,j} \\ &= \underbrace{\sum_{\substack{i \in \mathcal{U} \\ \varphi(i) = \{j\}}} c_{i,j}}_{\text{substitutions}} + \underbrace{\sum_{\substack{i \in \mathcal{U} \\ \varphi(i) = \{\epsilon\}}} c_{i,m+1}}_{\text{removals}} + \underbrace{\sum_{\substack{j \in \mathcal{V} \\ \varphi^{-1}[j] = \{\epsilon\}}} c_{n+1,j}}_{\text{insertions}} \\ &= \underbrace{\sum_{i \in \mathcal{U}^\#} c_{i, \tilde{\varphi}(i)}}_{\text{substitutions}} + \underbrace{\sum_{i \in \mathcal{U} \setminus \mathcal{U}^\#} c_{i,m+1}}_{\text{removals}} + \underbrace{\sum_{j \in \mathcal{V} \setminus \mathcal{V}^\#} c_{n+1,j}}_{\text{insertions}} \end{aligned} \quad (20)$$

**Problem 3 (LSAPE).** *Given to sets  $\mathcal{U}$  and  $\mathcal{V}$ , the linear sum assignment problem with edition (LSAPE) consists in finding an  $\epsilon$ -assignment having a minimal cost among all  $\epsilon$ -assignments transforming  $\mathcal{U}$  into  $\mathcal{V}$ , *i.e.* satisfying*

$$\operatorname{argmin}_{\varphi \in \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})} A_\epsilon(\varphi, \mathbf{C}). \quad (21)$$

This problem is equivalent to the sLSAPE, as demonstrated in the following section.

### 3.2 Link to the sLSAPE

Recall that the sLAPE finds a squared  $\epsilon$ -assignment from  $\mathcal{U}$  to  $\mathcal{V}$ , *i.e.* a bijection from  $\mathcal{U} \cup \mathcal{E}_\mathcal{U}$  to  $\mathcal{V} \cup \mathcal{E}_\mathcal{V}$ , with  $\mathcal{E}_\mathcal{U} = \{\epsilon_j, j \in \mathcal{V}\}$  and  $\mathcal{E}_\mathcal{V} = \{\epsilon_i, i \in \mathcal{U}\}$ , and such that  $i \in \mathcal{U}$  (resp.  $\epsilon_j \in \mathcal{E}_\mathcal{U}$ ) cannot be assigned to  $\epsilon_k \in \mathcal{E}_\mathcal{V}$  (resp.  $l \in \mathcal{V}$ ) with  $k \neq i$  (resp.  $l \neq j$ ). This is penalized through the cost values by setting a large value to such forbidden couplings. Also, the cost of assigning any element of  $\mathcal{E}_\mathcal{U}$  to any element of  $\mathcal{E}_\mathcal{V}$  is null (equivalent to  $\epsilon \rightarrow \epsilon$ , which has also a zero cost). We assume that any squared  $\epsilon$ -assignment has a non-infinite cost  $A_\epsilon$ .

**Lemma 1.** *Each  $\epsilon$ -assignment  $\varphi \in \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  is associated to a bijection  $\psi \in \mathcal{S}_\mathcal{E}(\mathcal{U}, \mathcal{V})$  (Problem 2) such that:*

$$\begin{cases} \forall i \in \mathcal{U}^\#, & \psi(i) = \varphi(i) \\ \forall i \in \mathcal{U} \setminus \mathcal{U}^\#, & \psi(i) = \epsilon_i \\ \forall j \in \mathcal{V} \setminus \mathcal{V}^\#, & \psi(\epsilon_j) = j \end{cases} \quad (22)$$

and the restriction of  $\psi$  to

$$\left( \begin{array}{l} \mathcal{E}_\mathcal{U}^\# = \{\epsilon_j \in \mathcal{E}_\mathcal{U} \mid j \in \mathcal{V}^\#\} \\ \epsilon_j \end{array} \mapsto \begin{array}{l} \mathcal{E}_\mathcal{V}^\# = \{\epsilon_i \in \mathcal{E}_\mathcal{V} \mid i \in \mathcal{U}^\#\} \\ \epsilon_i \end{array} \right) \quad (23)$$

Mapping	Initial set	Arrival set
$\psi^{\text{sub}} = \tilde{\varphi}$	$\mathcal{U}^\#$	$\mathcal{V}^\#$
$\psi^{\text{rem}}$	$\mathcal{U} \setminus \mathcal{U}^\#$	$\mathcal{E}_\mathcal{V} \setminus \mathcal{E}_\mathcal{V}^\#$
$\psi^{\text{ins}}$	$\mathcal{E}_\mathcal{U} \setminus \mathcal{E}_\mathcal{U}^\#$	$\mathcal{V} \setminus \mathcal{V}^\#$
$\psi^{\text{comp}}$	$\mathcal{E}_\mathcal{U}^\#$	$\mathcal{E}_\mathcal{V}^\#$

Table 1: Arrival and terminal sets involved in the construction of a squared  $\epsilon$ -assignment from an  $\epsilon$ -assignment.

is bijective. Moreover, mappings  $\varphi$  and  $\psi$  have the same cost (Eq. 5 and Eq. 20).

*Proof.* Let  $\varphi \in \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  be an  $\epsilon$ -assignment, and let  $\tilde{\varphi}$  be its associated injection. Consider the mapping  $\psi$  defined by Eq. 22 and Eq. 23. We show that  $\psi \in \mathcal{S}_\mathcal{E}(\mathcal{U}, \mathcal{V})$ . The mapping  $\psi: \mathcal{U} \cup \mathcal{E}_\mathcal{U} \rightarrow \mathcal{V} \cup \mathcal{E}_\mathcal{V}$  is built as described in Table 1:

- *Substitutions.* When  $i \in \mathcal{U}^\#$ , we have  $\psi(i) = \tilde{\varphi}(i)$ . Since  $\tilde{\varphi}$  is a bijection from  $\mathcal{U}^\#$  onto  $\mathcal{V}^\# = \tilde{\varphi}[\mathcal{U}^\#]$ , the restriction  $\psi^{\text{sub}}: \mathcal{U}^\# \rightarrow \psi[\mathcal{U}^\#] = \mathcal{V}^\#$  of  $\psi$  is also a bijection.
- *Removals/insertions.* When  $i \in \mathcal{U} \setminus \mathcal{U}^\#$  (reps.  $j \in \mathcal{V} \setminus \mathcal{V}^\#$ ), we have  $\psi(i) = \epsilon_i$  (resp.  $\psi(\epsilon_j) = j$ ). By definition, there is a natural bijective mapping from any  $i \in \mathcal{U}$  (resp.  $\epsilon_j \in \mathcal{E}_\mathcal{U}$ ) to  $\epsilon_i \in \mathcal{E}_\mathcal{V}$  (reps.  $j \in \mathcal{V}$ ). So the restriction  $\psi^{\text{rem}}: \mathcal{U} \setminus \mathcal{U}^\# \rightarrow \mathcal{E}_\mathcal{V} \setminus \mathcal{E}_\mathcal{V}^\#$  of  $\psi$ , and the restriction  $\psi^{\text{ins}}: \mathcal{E}_\mathcal{U} \setminus \mathcal{E}_\mathcal{U}^\# \rightarrow \mathcal{V} \setminus \mathcal{V}^\#$ , are bijective.
- *Completion.* Meanwhile all edit operations have been defined through the two last items, the mapping  $\psi$  is not yet complete. Indeed, it is not yet defined from  $\mathcal{E}_\mathcal{U}^\#$  to  $\mathcal{E}_\mathcal{V}^\#$ . Since  $\psi^{\text{sub}}$  is a bijection, we have  $|\mathcal{U}^\#| = |\mathcal{V}^\#|$  and by consequence  $|\mathcal{E}_\mathcal{U}^\#| = |\mathcal{E}_\mathcal{V}^\#|$ . So the restriction  $\psi^{\text{comp}}: \mathcal{E}_\mathcal{U}^\# \rightarrow \mathcal{E}_\mathcal{V}^\#$  of  $\psi$  can be any bijection from  $\mathcal{E}_\mathcal{U}^\#$  to  $\mathcal{E}_\mathcal{V}^\#$ .

All initial sets and arrival sets are disjoint and the mapping defined between each couple of sets is bijective. The mapping  $\psi$  is thus bijective. Moreover, considering that any mapping of  $\epsilon_j \in \mathcal{E}_\mathcal{U}^\#$  onto  $\psi^{\text{comp}}(\epsilon_j) \in \mathcal{E}_\mathcal{V}^\#$  is associated to a 0 cost, both mappings have the same cost.  $\square$

Note that one  $\epsilon$ -assignment corresponds to several bijections of  $\mathcal{S}_\mathcal{E}(\mathcal{U}, \mathcal{V})$  due to the bijective mapping  $\psi^{\text{comp}}$ . The proof also show that all these bijections have the same cost.

**Lemma 2.** Any bijection  $\psi \in \mathcal{S}_\mathcal{E}(\mathcal{U}, \mathcal{V})$  is associated to an  $\epsilon$ -assignment  $\varphi \in \mathcal{A}_\epsilon(\mathcal{U}, \mathcal{V})$  such that:

$$\begin{cases} \varphi(i) = \{\psi(i)\} & \text{if } \psi(i) \in \mathcal{V} \\ \varphi(i) = \{\epsilon\} & \text{if } \psi(i) \in \mathcal{E}_\mathcal{V} \\ \varphi(\epsilon) = \psi[\mathcal{E}_\mathcal{U}^\#] \cup \{\epsilon\} & \text{with } \mathcal{E}_\mathcal{U}^\# = \{\epsilon_j \in \mathcal{E}_\mathcal{U} \mid \psi(\epsilon_j) \in \mathcal{V}\} \end{cases} \quad (24)$$

Mappings  $\psi$  and  $\varphi$  have the same cost.

*Proof.* Let  $\psi \in \mathcal{S}_\mathcal{E}(\mathcal{U}, \mathcal{V})$  be a squared  $\epsilon$ -assignment. Since it defines a bijection from  $\mathcal{U} \cup \mathcal{E}_\mathcal{U}$  to  $\mathcal{V} \cup \mathcal{E}_\mathcal{V}$ , its restriction  $\tilde{\varphi}$  to  $\mathcal{U}^\# = \{u \in \mathcal{U} \mid \psi(u) \in \mathcal{V}\}$  is injective from  $\mathcal{U}^\#$  to  $\mathcal{V}$ . Such an injective mapping corresponds to a unique  $\epsilon$ -assignment from  $\mathcal{U}$  to  $\mathcal{V}$  (Proposition 1) whose cost is provided by Eq. 20 (with  $\mathcal{V}^\# = \tilde{\varphi}[\mathcal{U}^\#]$ ). By construction (Eq. 24),  $\varphi$  has the same cost as  $\psi$  (Eq. 5).  $\square$

A simple consequence of Lemma 1 and Lemma 2 is given by the following property.

**Proposition 2.** The LSAPÉ and the sLSAPÉ are equivalent, their respective solutions provide the same minimal cost  $A_\epsilon$ .

Note that even if the minimal cost is the same for the LSAPÉ and the sLSAPÉ, the proof of this proposition shows that solving a sLSAPÉ requires to define additionally a bijective mapping between two sets of  $\epsilon$ -values. This mapping being useless in terms of edit costs, computing an optimal  $\epsilon$ -assignment should be more efficient than its squared version.

$$\begin{matrix}
& (1,1) & \cdots & (1,m) & (1,\epsilon) & (2,1) & \cdots & (2,m) & (2,\epsilon) & \cdots & (n,1) & \cdots & (n,m) & (n,\epsilon) & (\epsilon,1) & \cdots & (\epsilon,m) \\
\begin{matrix} 1 \\ 2 \\ \vdots \\ n \\ 1 \\ \vdots \\ m \end{matrix} & \left( \begin{array}{cccc|cccc|c|cccc|c}
1 & \cdots & 1 & 1 & & & & & & & & & & & & & \\
& & & & & 1 & \cdots & 1 & 1 & & & & & & & & \\
& & & & & & & & & \cdots & & & & & & & \\
& & & & & & & & & & 1 & \cdots & 1 & 1 & & & \\
\hline
& & & & & 1 & & & & & 1 & & & & 1 & & \\
& & \ddots & & & & \ddots & & & & & \ddots & & & & \ddots & \\
& & & 1 & 0 & & & 1 & 0 & & & & 1 & 0 & & & 1
\end{array} \right)
\end{matrix}$$

Figure 2: Constraint matrix  $\mathbf{L}$ . Its  $n$  first rows represent  $\mathcal{U}$  and its  $m$  last rows represent  $\mathcal{V}$ . Missing values are equal to 0.

### 3.3 Matrix form and linear programming

Let  $\mathbf{C} \in [0, +\infty)^{(n+1) \times (m+1)}$  be an edit cost matrix. From Eq. 20, the cost associated to an  $\epsilon$ -assignment given by a matrix  $\mathbf{X} \in \mathcal{S}_{n,m,\epsilon}$  (Eq. 15 and Eq. 14), is measured by

$$A_\epsilon(\mathbf{X}, \mathbf{C}) = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} c_{i,j} x_{i,j}. \quad (25)$$

The LSAP (Problem 3) can then be rewritten as finding a matrix of  $\mathcal{S}_{n,m,\epsilon}$  having a minimal cost, *i.e.* satisfying

$$\operatorname{argmin}_{\mathbf{X} \in \mathcal{S}_{n,m,\epsilon}} A_\epsilon(\mathbf{X}, \mathbf{C}) \quad (26)$$

The LSAP can be rewritten as a linear programming problem, as described in the following.

Let  $\mathbf{x} = \operatorname{vec}(\mathbf{X}) \in \{0, 1\}^{(n+1)(m+1)-1}$  be the vectorization of  $\mathbf{X}$  obtained by concatenating its rows and by removing its last element (since  $x_{n+1,m+1} c_{n+1,m+1} = 0$ ). Then the constraints can be rewritten as the linear system of equations  $\mathbf{L}\mathbf{x} = \mathbf{1}$ , where the matrix  $\mathbf{L} \in \{0, 1\}^{(n+m) \times [(n+1)(m+1)-1]}$  is given by Fig. 2, *i.e.*

$$\forall(i, j), \quad \begin{cases} l_{k,(i,j)} = \delta_{k=i}, & \forall k = 1, \dots, n \\ l_{n+k,(i,j)} = \delta_{k=j}, & \forall k = 1, \dots, m \end{cases} \quad (27)$$

where  $\delta_{a=b} = 1$  if  $a=b$  and 0 else. The  $n$  first rows of  $\mathbf{L}$  represent the constraints on the rows of  $\mathbf{X}$  ( $|\varphi(i)| = 1, \forall i \in \mathcal{U}$ ) and its  $m$  last rows represent the constraints on the columns of  $\mathbf{X}$  ( $|\varphi^{-1}[j]| = 1, \forall j \in \mathcal{V}$ ). Each element  $l_{k,(i,j)} = 1$  corresponds to a possible edit operation:

- a substitution  $i \rightarrow j$ , when  $k = i \in \mathcal{U}$  or  $n+j$  with  $j \in \mathcal{V}$ ,
- a removal  $i \rightarrow \epsilon$  when  $k = i \in \mathcal{U}$  and  $j = m+1$ , or
- an insertion  $\epsilon \rightarrow j$  when  $k = n+j$  with  $j \in \mathcal{V}$ , and  $i = n+1$ .

Note that  $\epsilon$  is not represented as a row of  $\mathbf{L}$  since it is unconstrained. The matrix  $\mathbf{L}$  represents the discrete domain whereon solutions to the LSAP are defined.

Contrary to the constraint matrix involved in the LASP, which corresponds to the node-edge incidence matrix of the complete bipartite graph  $K_{n,m}$  with node sets  $\mathcal{U}$  and  $\mathcal{V}$ , the constraint matrix  $\mathbf{L}$  can be viewed as a node-edge incidence matrix of the mixed bipartite graph  $K_{n,m,\epsilon}$  composed of  $K_{n,m}$  and the two bipartite digraphs  $((\mathcal{V}, \{\epsilon\}), \mathcal{V} \times \{\epsilon\})$  and  $((\mathcal{U}, \{\epsilon\}), \mathcal{U} \times \{\epsilon\})$  representing insertion and removal operations, respectively. It is important to remark that  $K_{n,m,\epsilon}$  does not have any arc from  $\epsilon$  to  $\mathcal{U} \cup \mathcal{V}$ . Due to the definition of  $\mathbf{x}$ ,  $\mathbf{L}\mathbf{x}$  selects exactly one edge or arc in the neighborhood of each node in  $\mathcal{U} \cup \mathcal{V}$ . An  $\epsilon$ -assignment can thus be assimilated as a subgraph of  $K_{n,m,\epsilon}$  having each node in  $\mathcal{U} \cup \mathcal{V}$  connected to only one other node. Solving the LSAP consists in computing such a subgraph that minimizes the objective functional  $A_\epsilon$ .

Similarly, let  $\mathbf{c} = \text{vec}(\mathbf{C}) \in [0, +\infty)^{(n+1)(m+1)-1}$  be the vectorization of the edit cost matrix  $\mathbf{C}$ . With these notations, the LSAP given by Eq. 26 consists in selecting a vector satisfying

$$\text{argmin} \left\{ \mathbf{c}^T \mathbf{x} \mid \mathbf{L}\mathbf{x} = \mathbf{1}, \mathbf{x} \in \{0, 1\}^{(n+1)(m+1)-1} \right\} \quad (28)$$

which is a binary linear program.

By definition, each column of  $\mathbf{L}$  sums to no more than 2, and its rows can be partitioned into two sets,  $\mathcal{U}$  and  $\mathcal{V}$ , such that a 1 appears in each column at most once in each set (Fig. 2). Then the matrix  $\mathbf{L}$  is totally unimodular. By standard tools in linear programming, a binary linear programming problem with totally unimodular constraint matrix has always a binary optimal solution, see [18] for more details on linear programming problems. So the LSAP has a binary optimal solution. As for the LSAP (and the sLSAP), this shows that the LSAP can be solved with linear programming tools.

### 3.4 Primal-dual problem and admissible transformations

Consider the LSAP expressed as a binary linear program (Eq. 28). By standard tools in duality theory and linear programming [18], the problem dual to the LSAP consists in finding two vectors  $\mathbf{u} \in [0, +\infty)^n$  and  $\mathbf{v} \in [0, +\infty)^m$ , which associate a real value to each element of the sets  $\mathcal{U}$  and  $\mathcal{V}$  respectively, and satisfying

$$\text{argmax} \left\{ \mathbf{1}_n^T \mathbf{u} + \mathbf{1}_m^T \mathbf{v} \mid \mathbf{L}^T \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} \leq \mathbf{c}, \begin{pmatrix} \mathbf{u} \\ \mathbf{v} \end{pmatrix} \in [0, +\infty)^{n+m} \right\}$$

or equivalently

$$\begin{aligned} & \text{argmax}_{(\mathbf{u}, \mathbf{v})} \left\{ E(\mathbf{u}, \mathbf{v}) \stackrel{\text{def.}}{=} \mathbf{1}_n^T \mathbf{u} + \mathbf{1}_m^T \mathbf{v} \right\} \\ & \text{s.t. } \mathbf{u} \in [0, +\infty)^n, \mathbf{v} \in [0, +\infty)^m \\ & \quad u_i + v_j \leq c_{i,j}, \quad \forall i = 1, \dots, n, \quad \forall j = 1, \dots, m \\ & \quad u_i \leq c_{i,m+1}, \quad \forall i = 1, \dots, n \\ & \quad v_j \leq c_{n+1,j}, \quad \forall j = 1, \dots, m \end{aligned} \quad (29)$$

One can remark that the objective functional  $E(\mathbf{u}, \mathbf{v})$  is the same as the one involved in the problem dual to the LSAP, see [4], but the two last constraints are different.

By the strong duality theorem, if  $\mathbf{X}$  is a solution to the LSAP (primal problem), then its dual problem has an optimal solution  $(\mathbf{u}, \mathbf{v})$  and the solutions  $(\mathbf{X}, (\mathbf{u}, \mathbf{v}))$  satisfy

$$A_\epsilon(\mathbf{X}, \mathbf{C}) = E(\mathbf{u}, \mathbf{v}). \quad (30)$$

Such a primal-dual problem can be solved by finding transformations of the edit cost matrix, called admissible. We adapt the notion of admissible transformations of cost matrices (or equivalent cost matrices), developed in the context of the classical assignment problem to edit cost matrices, see for instance [4].

Note that the dual variables can be expressed as a pair  $(\mathbf{u}, \mathbf{v})$  such that  $\mathbf{u} \in [0, +\infty)^{n+1}$  with  $u_{n+1} = 0$ , and  $\mathbf{v} \in [0, +\infty)^{m+1}$  with  $v_{m+1} = 0$ , without altering objective functional  $E(\mathbf{u}, \mathbf{v})$  and leading to a rewriting of the three last constraints in Eq. 29 as

$$u_i + v_j \leq c_{i,j}, \quad \forall i = 1, \dots, n+1, \quad j = 1, \dots, m+1. \quad (31)$$

We use this trick to simplify the forthcoming expressions.

**Definition 3** ( $\epsilon$ -admissible transformation). *Let  $T(\mathbf{C}, \overline{\mathbf{C}})$  be the transformation of an edit cost matrix  $\mathbf{C}$  into a matrix  $\overline{\mathbf{C}}$ , with  $\mathbf{C}, \overline{\mathbf{C}} \in [0, +\infty)^{(n+1) \times (m+1)}$ .  $T$  is called  $\epsilon$ -admissible with index  $A(T) \in \mathbb{R}$  if  $\overline{\mathbf{C}}$  is an edit cost matrix and*

$$A_\epsilon(\mathbf{X}, \mathbf{C}) = A_\epsilon(\mathbf{X}, \overline{\mathbf{C}}) + A(T)$$

for any  $\epsilon$ -assignment  $\mathbf{X}$ .

By consequence, an  $\epsilon$ -assignment  $\mathbf{X}$  which minimizes  $A_\epsilon(\mathbf{X}, \mathbf{C})$  also minimizes  $A_\epsilon(\mathbf{X}, \overline{\mathbf{C}}) + A(T)$ , and so the edit cost matrices  $\mathbf{C}$  and  $\overline{\mathbf{C}}$  are said to be equivalent (w.r.t. the LSAP). Remark that transformations are assumed to produce edit cost matrices. In particular, transformed matrices must satisfy  $\overline{\mathbf{C}} \geq \mathbf{0}$ .

**Lemma 3.** *Let  $T(\mathbf{C}, \overline{\mathbf{C}})$  be an  $\epsilon$ -admissible transformation. Let  $\hat{\mathbf{X}}$  be an  $\epsilon$ -assignment satisfying  $A_\epsilon(\hat{\mathbf{X}}, \overline{\mathbf{C}}) = 0$ . Then  $\hat{\mathbf{X}}$  represents an optimal  $\epsilon$ -assignment, with total cost  $A_\epsilon(\hat{\mathbf{X}}, \mathbf{C}) = A(T)$ .*

*Proof.* Let  $\mathbf{X}$  be an arbitrary  $\epsilon$ -assignment. Under the conditions of the lemma, we have  $\overline{\mathbf{C}} \geq \mathbf{0}$  and then  $A_\epsilon(\mathbf{X}, \overline{\mathbf{C}}) \geq 0$ . So  $A_\epsilon(\mathbf{X}, \mathbf{C}) = A_\epsilon(\mathbf{X}, \overline{\mathbf{C}}) + A(T) \geq A(T) = A_\epsilon(\hat{\mathbf{X}}, \overline{\mathbf{C}}) + A(T) = A_\epsilon(\hat{\mathbf{X}}, \mathbf{C})$ , and  $A_\epsilon(\mathbf{X}, \mathbf{C})$  reaches its minimum value for  $\mathbf{X} = \hat{\mathbf{X}}$ .  $\square$

Admissible transformations of edit cost matrices are linked to the primal-dual equation 30 as follows.

**Proposition 3.** *Let  $\mathbf{C}$  be an edit cost matrix. Let  $\mathbf{a} \in \mathbb{R}^{n+1}$  and  $\mathbf{b} \in \mathbb{R}^{m+1}$  be two vectors such that  $a_{n+1} = b_{m+1} = 0$  and  $\mathbf{a}\mathbf{1}_{m+1}^T + \mathbf{1}_{n+1}\mathbf{b}^T \leq \mathbf{C}$ . The transformation  $T(\mathbf{C}, \overline{\mathbf{C}})$  such that  $\overline{\mathbf{C}} = \mathbf{C} - \mathbf{a}\mathbf{1}_{m+1}^T - \mathbf{1}_{n+1}\mathbf{b}^T$  is  $\epsilon$ -admissible with index  $A(T) = E(\mathbf{a}, \mathbf{b})$ .*

*Proof.* Let  $\mathbf{X}$  be any  $\epsilon$ -assignment. We have:

$$\begin{aligned} A_\epsilon(\mathbf{X}, \overline{\mathbf{C}}) &= \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} (c_{i,j} - a_i - b_j)x_{i,j} = A_\epsilon(\mathbf{X}, \mathbf{C}) - \sum_{i=1}^n a_i \underbrace{\sum_{j=1}^{m+1} x_{i,j}}_{=1} - \sum_{j=1}^m b_j \underbrace{\sum_{i=1}^{n+1} x_{i,j}}_{=1} \\ &= A_\epsilon(\mathbf{X}, \mathbf{C}) - (\mathbf{1}_{n+1}^T \mathbf{a} + \mathbf{1}_{m+1}^T \mathbf{b}) = A_\epsilon(\mathbf{X}, \mathbf{C}) - E(\mathbf{a}, \mathbf{b}). \end{aligned}$$

So  $A_\epsilon(\mathbf{X}, \mathbf{C}) = A_\epsilon(\mathbf{X}, \overline{\mathbf{C}}) + E(\mathbf{a}, \mathbf{b})$ . Remark that  $\overline{\mathbf{C}} \geq \mathbf{0}$  with  $c_{n+1, m+1} = 0$ , so  $\overline{\mathbf{C}}$  is an edit cost matrix and the transformation is  $\epsilon$ -admissible.  $\square$

Consider a dual solution  $(\mathbf{u}, \mathbf{v})$ . Recall that  $u_{n+1} = v_{m+1} = 0$ . By definition the dual solution fulfils the conditions of Proposition 3. This implies the following complementary slackness condition.

**Proposition 4.**  *$(\mathbf{X}, (\mathbf{u}, \mathbf{v}))$  solves the LSAP and its dual problem (primal-dual equation Eq. 30) iff  $A_\epsilon(\mathbf{X}, \mathbf{C}) = 0$  with*

$$\overline{\mathbf{C}} = \mathbf{C} - \mathbf{u}\mathbf{1}_{m+1}^T - \mathbf{1}_{n+1}\mathbf{v}^T$$

*i.e. for all  $(i, j) \in \{1, \dots, n+1\} \times \{1, \dots, m+1\}$*

$$\begin{aligned} x_{i,j}\bar{c}_{i,j} = 0 &\Leftrightarrow ((x_{i,j} = 1) \wedge (\bar{c}_{i,j} = 0)) \vee ((x_{i,j} = 0) \wedge (\bar{c}_{i,j} \geq 0)) \\ &\Leftrightarrow ((x_{i,j} = 1) \wedge (c_{i,j} = u_i + v_j)) \vee ((x_{i,j} = 0) \wedge (c_{i,j} \geq u_i + v_j)) \end{aligned} \quad (32)$$

*The optimal cost is  $A_\epsilon(\mathbf{X}, \mathbf{C}) = E(\mathbf{u}, \mathbf{v})$ .*

*Proof.* Suppose that  $(\mathbf{X}, (\mathbf{u}, \mathbf{v}))$  is a solution to the primal-dual equation 30. By Eq. 31 we have  $\mathbf{u}\mathbf{1}^T + \mathbf{1}\mathbf{v}^T \leq \mathbf{C}$ . Then by Proposition 3,  $\overline{\mathbf{C}} = \mathbf{C} - (\mathbf{u}\mathbf{1}^T + \mathbf{1}\mathbf{v}^T)$  is an  $\epsilon$ -admissible transformation:  $A_\epsilon(\mathbf{X}, \mathbf{C}) = A_\epsilon(\mathbf{X}, \overline{\mathbf{C}}) + E(\mathbf{u}, \mathbf{v})$ . Since  $A_\epsilon(\mathbf{X}, \mathbf{C}) = E(\mathbf{u}, \mathbf{v})$  by Eq. 30, so  $A_\epsilon(\mathbf{X}, \overline{\mathbf{C}}) = 0$ , which demonstrates the first part ( $\Rightarrow$ ). Now suppose that  $\mathbf{X}$  is an  $\epsilon$ -assignment, and  $\overline{\mathbf{C}}$  is an edit cost matrix such that  $\overline{\mathbf{C}} = \mathbf{C} - (\mathbf{u}\mathbf{1}^T + \mathbf{1}\mathbf{v}^T)$  and  $A_\epsilon(\mathbf{X}, \overline{\mathbf{C}}) = 0$ . By Lemma 3  $\mathbf{X}$  is optimal with cost  $A_\epsilon(\mathbf{X}, \mathbf{C}) = E(\mathbf{u}, \mathbf{v})$ .  $\square$

This condition is the same for the LSAP, but  $\mathbf{X}$  and  $\mathbf{C}$  do not satisfy the same constraints. Solving the LSAP is equivalent to find a transformed cost matrix having  $k$  independent zero elements ( $k = \min\{n, m\}$ ). Elements of a matrix are independent if none of them occupies the same row or column. According to König's theorem and Eq. 32, this independent set of zero elements exactly corresponds to an optimal assignment [8, 11, 1]. Hungarian-type algorithms are based on this property. In order to derive similar algorithms for solving the LSAP, the notion of independent elements needs to be adapted.

**Definition 4** ( $\epsilon$ -independent). Let  $\mathcal{S}$  be a set of elements of a  $(n+1) \times (m+1)$  edit cost matrix  $\mathbf{C}$ . Elements of  $\mathcal{S}$  are  $\epsilon$ -independent if there is at most one element of  $\mathcal{S}$  on each row  $i=1, \dots, n$  of  $\mathbf{C}$ , and at most one element of  $\mathcal{S}$  on each column  $j=1, \dots, m$  of  $\mathbf{C}$ . The set  $\mathcal{S}$  is maximal when there is exactly one element of  $\mathcal{S}$  on each row  $i=1, \dots, n$ , and exactly one element of  $\mathcal{S}$  on each column  $j=1, \dots, m$ .

Obviously, by definition, any  $\epsilon$ -assignment contains a maximal set of  $\epsilon$ -independent elements equals to 1. This implies the following property.

**Corollary 3.** Let  $\mathcal{S}$  be a maximal set of  $\epsilon$ -independent elements of an edit cost matrix  $\mathbf{C}$ . Let  $\mathbf{u}$  and  $\mathbf{v}$  be two vectors such that

$$\begin{cases} c_{i,j} = u_i + v_j, & \forall (i,j) \in \mathcal{S}, \\ c_{i,j} \geq u_i + v_j, & \forall (i,j) \notin \mathcal{S}. \end{cases} \quad (33)$$

Then the matrix  $\mathbf{X}$  constructed such that  $x_{i,j} = \delta_{(i,j) \in \mathcal{S}}$  is an optimal  $\epsilon$ -assignment for the LSAP, and  $(\mathbf{u}, \mathbf{v})$  solves its dual problem.

*Proof.* By construction and from Definition 4,  $\mathbf{X}$  defines an  $\epsilon$ -assignment. Also,  $\mathbf{X}$  and  $\mathbf{C}$  satisfy the complementary slackness condition given by Eq. 32. So by Proposition 4  $(\mathbf{X}, (\mathbf{u}, \mathbf{v}))$  solves the LSAP and its dual problem.  $\square$

The Hungarian-type algorithm proposed in the following section finds such a maximal set of  $\epsilon$ -independent elements.

## 4 A Hungarian algorithm for solving the LSAP

We propose to solve the LSAP (Problem 3), and its dual problem (Eq. 29), by adapting the Hungarian algorithm that initially solves the LSAP and its dual problem. Contrary to the classical Khun-Munkres algorithm [8, 11, 1], that transforms the cost matrix, this version of the Hungarian algorithm updates equivalently the dual variables until a maximal set of independent elements is obtained, which is known to be more efficient both in time and space [10, 4].

We consider an edit cost matrix  $\mathbf{C} \in [0, +\infty)^{(n+1) \times (m+1)}$ , as defined by Eq. 19, between two sets  $\mathcal{U} = \{1, \dots, n\}$  and  $\mathcal{V} = \{1, \dots, m\}$ . Recall that row  $n+1$  of  $\mathbf{C}$  represents the cost of inserting each element of  $\mathcal{V}$  into  $\mathcal{U}$ . Similarly, column  $m+1$  represents the cost of removing each element of  $\mathcal{U}$ . The adaptation of the Hungarian algorithm takes into account the relaxation of the constraints associated to removal and insertion operations. Intuitively, the algorithm first computes an initial pair of dual variables and an associated partial  $\epsilon$ -assignment (Section 2.3). Then, while there exists an element  $e$  of  $\mathcal{V}$  that is not assigned to an element of  $\mathcal{U}_e$ , a new partial  $\epsilon$ -assignment is constructed such that  $e$  is assigned, and all previous assigned elements are still assigned. Since all elements of  $\mathcal{U}$  may not be assigned, the algorithm proceed similarly until they are. Each iteration corresponds to solve a restricted primal-dual problem, *i.e.* finding a set of  $\epsilon$ -independent elements.

### 4.1 Encoding and pre-processing

In the following algorithms, a partial  $\epsilon$ -assignment is not encoded by its associated binary matrix  $\mathbf{X} \in \{0, 1\}^{(n+1) \times (m+1)}$  (Section 2.3) but by a pair of vectors  $(\rho, \varrho)$  with  $\rho \in \{0, 1, \dots, m+1\}^n$ ,  $\varrho \in \{0, 1, \dots, n+1\}^m$ , and such that

$$\begin{aligned} \rho_i &= \begin{cases} j & \text{if } \exists j \in \{1, \dots, m+1\} \mid x_{i,j} = 1 \\ 0 & \text{else (unassigned)} \end{cases} \\ \varrho_j &= \begin{cases} i & \text{if } \exists i \in \{1, \dots, n+1\} \mid x_{i,j} = 1 \\ 0 & \text{else (unassigned)} \end{cases} \end{aligned} \quad (34)$$

Note that unassigned elements of the partial  $\epsilon$ -assignment are assigned to 0. The matrix  $\mathbf{X}$  can be easily reconstructed from  $\rho$  and  $\varrho$ .

Let  $\mathbf{C} \in [0, +\infty)^{(n+1) \times (m+1)}$  be an edit cost matrix. We want to compute an initial partial  $\epsilon$ -assignment  $\mathbf{X}$ , and a pair of dual variables  $(\mathbf{u}, \mathbf{v})$ , such that Eq. 32 is satisfied. To this

```

1 begin PreProcessing(C)
2    $u_{n+1}, v_{m+1} \leftarrow 0, \rho \leftarrow \mathbf{0}_n, \varrho \leftarrow \mathbf{0}_m$ 
3   for  $i=1, \dots, n$  do  $u_i \leftarrow \min \{c_{i,j} \mid j=1, \dots, m+1\}$ 
4   for  $j=1, \dots, m$  do  $v_j \leftarrow \min \{c_{i,j} - u_i \mid i=1, \dots, n+1\}$ 
5   for  $i=1, \dots, n$  do
6     for  $j=1, \dots, m$  do
7       if  $(\varrho_j = 0) \wedge (c_{i,j} = u_i + v_j)$  then
8          $\rho_i \leftarrow j, \varrho_j \leftarrow i$  //  $i$  assigned to  $j$ 
9         break
10    if  $(\rho_i = 0) \wedge (c_{i,m+1} = u_i)$  then  $\rho_i \leftarrow m+1$  //  $i \rightarrow \epsilon$ 
11    for  $j=1, \dots, m$  do
12      if  $(\varrho_j = 0) \wedge (c_{n+1,j} = v_j)$  then  $\varrho_j \leftarrow n+1$  //  $\epsilon \rightarrow j$ 
13  return  $((\rho, \varrho), (\mathbf{u}, \mathbf{v}))$ 

```

**Algorithm 1:** Basic pre-processing

we adapt the basic pre-processing step involved in the Hungarian algorithm [4]. Algorithm 1 details this step. The minimum cost on each row  $i \in \mathcal{U}$  of  $\mathbf{C}$  allows to construct  $\mathbf{u}$ . Then, the minimum cost on each column  $j \in \mathcal{V}$  of  $\mathbf{C} - \mathbf{u}\mathbf{1}^T$  allows to construct  $\mathbf{v}$ . By construction we have  $\mathbf{C} \geq \mathbf{u}\mathbf{1}^T + \mathbf{1}\mathbf{v}^T$ . Let  $\bar{\mathbf{C}} = \mathbf{C} - \mathbf{u}\mathbf{1}^T - \mathbf{1}\mathbf{v}^T$  be the transformed edit cost matrix (Proposition 3). This transformation guaranties the existence of at least one element  $j \in \mathcal{V}$  satisfying  $c_{i,j} = u_i + v_j$ , or equivalently  $\bar{c}_{i,j} = 0$ , for each  $i \in \mathcal{U}$ . In other terms, each row  $i \in \mathcal{U}$  of  $\bar{\mathbf{C}}$  contains at least one 0. Similarly, each column  $j \in \mathcal{V}$  of  $\bar{\mathbf{C}}$  contains at least one 0. So it is possible to deduce a partial  $\epsilon$ -assignment, eventually reduced to one pair of assigned elements only, satisfying Eq. 32. Algorithm 1 finds such a set by a scanline approach. Note that the whole algorithm has a  $O(nm)$  time complexity.

Algorithm 1 has a  $O(nm)$  time complexity. More sophisticated pre-processing methods can be adapted to find an initial partial  $\epsilon$ -assignment, see [4] for more details on these methods.

**Example 1.** Consider the edit cost matrix

$$\mathbf{C} = \left( \begin{array}{ccccc|c} 7 & 11 & 9 & 8 & 9 & 10 \\ 2 & 8 & 8 & 5 & 7 & 3 \\ 1 & 7 & 6 & 6 & 9 & 5 \\ 3 & 7 & 6 & 2 & 2 & 3 \\ \hline 4 & 2 & 2 & 7 & 8 & 0 \end{array} \right) \quad (35)$$

First part of Algorithm 1 computes

$$\mathbf{u} = \begin{pmatrix} 7 \\ 2 \\ 1 \\ 2 \\ 0 \end{pmatrix} \text{ and } \mathbf{v} = \begin{pmatrix} 0 \\ 2 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \text{ so } \bar{\mathbf{C}} = \mathbf{C} - \mathbf{u}\mathbf{1}^T - \mathbf{1}\mathbf{v}^T = \left( \begin{array}{ccccc|c} 0 & 2 & 0 & 1 & 2 & 3 \\ 0 & 4 & 4 & 3 & 5 & 1 \\ 0 & 4 & 3 & 5 & 8 & 4 \\ 1 & 3 & 2 & 0 & 0 & 1 \\ \hline 4 & 0 & 0 & 7 & 8 & 0 \end{array} \right).$$

Due to the distribution of the zeros in  $\bar{\mathbf{C}}$ , which correspond to the elements of  $\mathbf{C}$  satisfying  $c_{i,j} = u_i + v_j$ , only partial assignments with edition can be constructed. Second part of Algorithm 1 leads to  $\rho = (1, 0, 0, 4)$  and  $\varrho = (1, 5, 5, 4, 0)$ , equivalently represented by

$$\bar{\mathbf{C}} = \left( \begin{array}{ccccc|c} 0^* & 2 & 0 & 1 & 2 & 3 \\ 0 & 4 & 4 & 3 & 5 & 1 \\ 0 & 4 & 3 & 5 & 8 & 4 \\ 1 & 3 & 2 & 0^* & 0 & 1 \\ \hline 4 & 0^* & 0^* & 7 & 8 & 0 \end{array} \right),$$

where  $\bar{c}_{i,j} = 0^*$  corresponds to  $i \rightarrow j$  in the partial  $\epsilon$ -assignment.



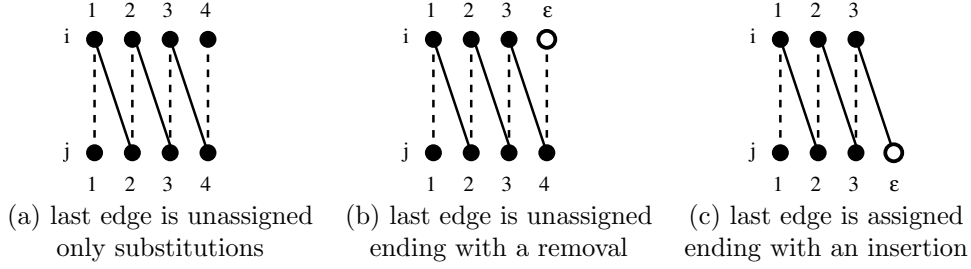


Figure 3: Three cases of augmenting paths starting with an unassigned edge  $(i_1, j_1)$ . Unassigned edges are dotted.

## 4.2 Augmenting paths

Consider a partial  $\epsilon$ -assignment  $(\rho, \varrho)$  (or equivalently  $\mathbf{X}$ ) and a pair of dual variables  $(\mathbf{u}, \mathbf{v})$  satisfying Eq. 32. The partial  $\epsilon$ -assignment may be empty (no pre-processing), *i.e.*  $\rho = \mathbf{0}_n$ ,  $\varrho = \mathbf{0}_m$ ,  $\mathbf{u} = \mathbf{0}_{n+1}$  and  $\mathbf{v} = \mathbf{0}_{m+1}$ . Assume that at least one element  $k \in \mathcal{V}$  is not assigned to an element of  $\mathcal{U}_\epsilon$ , for instance  $k = 5 \in \mathcal{V}$  in Example 1. We want to find a new partial  $\epsilon$ -assignment such that:

1.  $k$  becomes assigned to an element of  $\mathcal{U}_\epsilon$ ,
2. all previously assigned elements are still assigned, and
3. there is a pair of dual variables such that Eq. 32 is satisfied.

Let  $\mathcal{S}_{n,m,\epsilon,k}$  be the set of all matrices representing a partial  $\epsilon$ -assignment which satisfies constraints 1 and 2 above.

To solve this sub-problem of the LSAP, we consider subgraph of the mixed bipartite graph  $K_{n,m,\epsilon}$  (Section 3.3) whose edges/arcs satisfies  $c_{i,j} = u_i + v_j$ , *i.e.*  $\bar{c}_{i,j} = 0$ . Let  $G^0$  be this subgraph. By definition, the bipartite graph which represents the current partial  $\epsilon$ -assignment  $\mathbf{X}$  is a subgraph of  $G^0$  (its node adjacency matrix is  $\mathbf{X}$ ). It defines a set of assigned nodes and edges on  $G^0$ , an edge  $(i, j)$  being assigned iff  $x_{i,j} = 1$  in the current partial  $\epsilon$ -assignment.

The new partial  $\epsilon$ -assignment is constructed by computing minimal paths in  $K_{n,m,\epsilon}$ , *i.e.* paths in  $G^0$ . By definition, a path  $P$  in  $K_{n,m,\epsilon}$  alternate an element of a set and an element of the other. Also, since there is no arc from  $\epsilon$  to any other element of  $\mathcal{U} \cup \mathcal{V}$  in  $K_{n,m,\epsilon}$  (Section 3.3),  $\epsilon$  can only be a terminal node of a path. Its length, penalized by the edit costs, is defined by

$$\gamma_{\mathcal{C}}(P) = \sum_{(i,j) \subset P} c_{i,j}. \quad (36)$$

**Definition 5** (alternating and augmenting paths). *A path in  $K_{n,m,\epsilon}$  is alternating if its edges/arcs are alternatively unassigned and assigned. An alternating path, which starts with an unassigned element  $k$  of  $\mathcal{V}$  (resp.  $\mathcal{U}$ ) and ends with*

1. *an unassigned element of  $\mathcal{U}$  (resp.  $\mathcal{V}$ ), or*
2. *the null element  $\epsilon$  of  $\mathcal{U}_\epsilon$  or  $\mathcal{V}_\epsilon$ ,*

*is an augmenting path if its length defined by Eq. 36 is minimal among all the alternating paths connecting  $k$  to an element satisfying one of the cases above.*

Contrary to the classical definition of augmenting paths, see [4], an augmenting path as defined above may end with an assigned arc. Assume that the starting node of the path is  $k \in \mathcal{V}$ . Then, these two configurations can be encountered:

- If  $i_l \in \mathcal{U}_\epsilon$  is the last node, the path can be represented as a sequence  $(k = j_1, i_1, j_2, i_2, \dots, j_l, i_l)$  with  $j_s \in \mathcal{V}$  and  $i_r \in \mathcal{U}$  for  $r = 1, \dots, l-1$ . Since the first edge  $(j_1, i_1)$  is unassigned and since the path is alternating, the last edge  $(j_l, i_l)$  is also unassigned (Fig. 4.2(a,b)), as in the classical case.

- If  $\epsilon \in \mathcal{V}_\epsilon$  is the last node, the path can be represented as a sequence  $(k = j_1, i_1, j_2, i_2, \dots, j_l, i_l, \epsilon)$  with  $j_s \in \mathcal{V}$  and  $i_r \in \mathcal{U}$ . In this case the last arc  $(i_l, \epsilon)$  is assigned, i.e.  $i_l$  is assigned to  $\epsilon$  (Fig. 4.2(c)).

Once an augmenting path has been found, a new partial  $\epsilon$ -assignment is constructed by assigning unassigned edges/arcs of the path, and reciprocally, as follows.

**Proposition 5.** *Let  $\mathbf{X}$  be the matrix representation of a partial  $\epsilon$ -assignment, and  $(\mathbf{u}, \mathbf{v})$  be a pair of dual variables ( $u_{n+1} = v_{m+1} = 0$ ), such that the complementary slackness condition (Eq. 32) is satisfied. Let  $P$  be an augmenting path starting with an unassigned element  $k \in \mathcal{V}$  (resp.  $\mathcal{U}$ ). The matrix  $\mathbf{X}'$ , defined by*

$$x'_{i,j} = \begin{cases} 1 - x_{i,j} & \text{if } (i,j) \in P \\ x_{i,j} & \text{else} \end{cases},$$

represents an optimal partial  $\epsilon$ -assignment, i.e. satisfying

$$\mathbf{X}' \in \operatorname{argmin} \{A_\epsilon(\mathbf{Y}, \mathbf{C}) \mid \mathbf{Y} \in \mathcal{S}_{n,m,\epsilon,k}\}, \quad (37)$$

Its total cost is equal to  $A_\epsilon(\mathbf{X}', \mathbf{C}) = \gamma_{\bar{\mathbf{C}}}(P) + E(\mathbf{u}, \mathbf{v})$ , with  $\bar{\mathbf{C}} = \mathbf{C} - \mathbf{u}\mathbf{1}_{m+1}^T - \mathbf{1}_{n+1}\mathbf{v}^T$ .

*Proof.* Let  $P$  be an augmenting path starting with an element  $k \in \mathcal{V}$ . We have  $\mathbf{X}' \in \{0, 1\}^{(n+1) \times (m+1)}$  by definition. We show that any column  $j \in \mathcal{V}$  of  $\mathbf{X}'$  sums to no more than 1. First remark that for any  $j$  included in  $P$  we have:

$$\sum_{i=1}^{n+1} x'_{i,j} = \sum_{(i,j) \in P} x'_{i,j} + \sum_{(i,j) \notin P} x'_{i,j} = \sum_{(i,j) \in P} (1 - x_{i,j}) + \sum_{(i,j) \notin P} x_{i,j} = \sum_{(i,j) \in P} (1 - x_{i,j})$$

since there is at most one  $i \in \mathcal{U}_\epsilon$  such that  $x_{i,j} = 1$ , and this element is included in  $P$ . Moreover we have:

- For  $j = k$ , there is no  $i \in \mathcal{U}_\epsilon$  such that  $x_{i,k} = 1$ , then the neighbor  $i_1$  of  $k$  in  $P$  satisfies  $x_{i_1,k} = 0$ . So  $x'_{i_1,k} = 1$  and  $x'_{i,k} = 0$  for all  $i \in \mathcal{U}_\epsilon \setminus \{i_1\}$ .
- For any  $j \in \mathcal{V} \setminus \{k\}$  included in  $P$ ,  $j$  has two neighbors  $i_a$  and  $i_b$  of  $\mathcal{U}_\epsilon$  in  $P$  satisfying for instance  $x_{i_a,j} = 1$  and  $x_{i_b,j} = 0$ . So we have  $x'_{i_a,j} = 0$ ,  $x'_{i_b,j} = 1$ , and  $x'_{i,j} = 0$  for all  $i \in \mathcal{U}_\epsilon \setminus \{i_a, i_b\}$ .

So  $\sum_{i=1}^{n+1} x'_{i,j} \in \{0, 1\}$  for any  $j$  included in  $P$ . When  $j$  is not a node of  $P$ , we have  $\sum_{i=1}^{n+1} x'_{i,j} = \sum_{i=1}^{n+1} x_{i,j} \in \{0, 1\}$  by definition.

Similarly, we have  $\sum_{j=1}^{m+1} x'_{i,j} \in \{0, 1\}$  and so  $\mathbf{X}'$  is a partial  $\epsilon$ -assignment. This shows that any augmenting path starting with an unassigned element  $k \in \mathcal{V}$  leads to a new partial  $\epsilon$ -assignment that assigns this element. Now we show that this partial  $\epsilon$ -assignment is optimal among all partial  $\epsilon$ -assignments in  $\mathcal{S}_{n,m,\epsilon,k}$ .

Since  $\mathbf{X}$  and  $\bar{\mathbf{C}}$  satisfy the complementary slackness condition, the total cost  $A_\epsilon(\mathbf{X}, \bar{\mathbf{C}})$  can be decomposed into 4 terms:

$$A_\epsilon(\mathbf{X}, \bar{\mathbf{C}}) = \underbrace{\sum_{\substack{(i,j) \in P \\ x_{i,j}=1}} \bar{c}_{i,j} x_{i,j}}_{=0 \quad (\bar{c}_{i,j}=0)} + \underbrace{\sum_{\substack{(i,j) \in P \\ x_{i,j}=0}} \bar{c}_{i,j} x_{i,j}}_{=0 \quad (\bar{c}_{i,j} \geq 0)} + \underbrace{\sum_{\substack{(i,j) \notin P \\ x_{i,j}=1}} \bar{c}_{i,j} x_{i,j}}_{=0 \quad (\bar{c}_{i,j}=0)} + \underbrace{\sum_{\substack{(i,j) \notin P \\ x_{i,j}=0}} \bar{c}_{i,j} x_{i,j}}_{=0 \quad (\bar{c}_{i,j} \geq 0)}$$

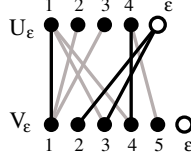
Similarly we have for  $\mathbf{X}'$ :

$$\begin{aligned} A_\epsilon(\mathbf{X}', \bar{\mathbf{C}}) &= \underbrace{\sum_{\substack{(i,j) \in P \\ x_{i,j}=1}} \bar{c}_{i,j} (1 - x_{i,j})}_{=0 \quad (\bar{c}_{i,j}=0)} + \underbrace{\sum_{\substack{(i,j) \in P \\ x_{i,j}=0}} \bar{c}_{i,j} (1 - x_{i,j})}_{\geq 0 \quad (\bar{c}_{i,j} \geq 0)} + \underbrace{\sum_{\substack{(i,j) \notin P \\ x_{i,j}=1}} \bar{c}_{i,j} x_{i,j}}_{=0 \quad (\bar{c}_{i,j}=0)} + \underbrace{\sum_{\substack{(i,j) \notin P \\ x_{i,j}=0}} \bar{c}_{i,j} x_{i,j}}_{=0 \quad (\bar{c}_{i,j} \geq 0)} \\ &= 0 + \underbrace{\sum_{\substack{(i,j) \in P \\ x_{i,j}=0}} \bar{c}_{i,j}}_{\geq 0 \quad (\bar{c}_{i,j} \geq 0)} + 0 + 0 = \gamma_{\bar{\mathbf{C}}}(P) \end{aligned}$$

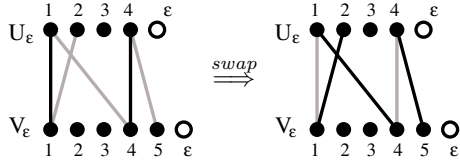
Since  $P$  is a minimal path among all augmenting paths connecting  $k$  to an unassigned element of  $\mathcal{U}$  or to  $\epsilon$ ,  $\mathbf{X}'$  is a partial  $\epsilon$ -assignment with a minimal cost.  $\square$

If each edge/arc of  $P$  satisfies  $\bar{c}_{i,j}=0$  ( $P$  is thus contained in  $G^0$ ), then  $\gamma_{\bar{\mathbf{C}}}(P)=0$  and by consequence  $A_\epsilon(\mathbf{X}', \mathbf{C})=E(\mathbf{u}, \mathbf{v})$ . The sub-problem (Eq. 37) of the LSAPE can thus be solved by computing an augmenting path  $P$ , together with a transformed edit cost matrix  $\bar{\mathbf{C}}$ , such that  $\gamma_{\bar{\mathbf{C}}}(P)=0$ . This is detailed in Section 4.3. The new partial  $\epsilon$ -assignment is then obtained by assigning unassigned edges and by deassigning assigned edges along the path (Proposition 5), as detailed in Section 4.4.

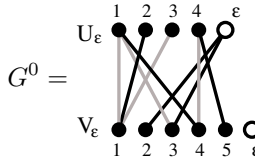
**Example 2.** Consider the partial  $\epsilon$ -assignment defined by  $\rho=(1, 0, 0, 4)$  and  $\varrho=(1, 5, 5, 4, 0)$  for the edit cost matrix

$$\left( \begin{array}{ccccc|c} \mathbf{0}^* & 2 & 0 & 0 & 1 & 3 \\ 0 & 4 & 4 & 2 & 4 & 1 \\ 0 & 4 & 3 & 4 & 7 & 4 \\ 2 & 4 & 3 & \mathbf{0}^* & 0 & 2 \\ \hline 4 & \mathbf{0}^* & \mathbf{0}^* & 6 & 7 & 0 \end{array} \right), \quad G^0 =$$


We want to assign  $k=5 \in \mathcal{V}$ . In the corresponding bipartite graph  $G^0$  above (gray and black edges),  $k=5$  can be linked to an unassigned element of  $\mathcal{U}$  (2 or 3), by a path which alternates unassigned (gray) and assigned (black) edges, i.e. an augmenting path. Consider the one ending with  $2 \in \mathcal{U}$ :



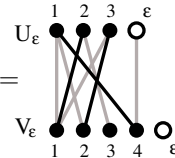
By swapping assigned and unassigned edges along the path:

$$G^0 = \left( \begin{array}{ccccc|c} \mathbf{0} & 2 & 0 & \mathbf{0}^* & 1 & 3 \\ \mathbf{0}^* & 4 & 4 & 2 & 4 & 1 \\ 0 & 4 & 3 & 4 & 7 & 4 \\ 2 & 4 & 3 & \mathbf{0} & \mathbf{0}^* & 2 \\ \hline 4 & \mathbf{0}^* & \mathbf{0}^* & 6 & 7 & 0 \end{array} \right)$$


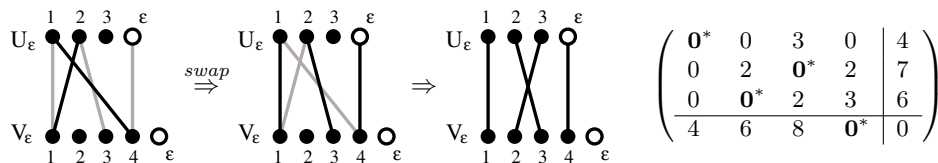
we obtain the partial  $\epsilon$ -assignment  $\rho=(4, 1, 0, 5)$  and  $\varrho=(2, 5, 5, 1, 4)$ .

The augmenting path can also ends with an element  $\epsilon$ , either  $(n+1) \in \mathcal{U}_\epsilon$  or  $(m+1) \in \mathcal{V}_\epsilon$ . In one of these cases and contrary to the classical definition of augmenting paths [4], an augmenting path can end with an assigned arc, as illustrated in the following example.

**Example 3.** Consider the partial  $\epsilon$ -assignment defined by  $\rho=(4, 1, 2)$  and  $\varrho=(2, 3, 0, 1)$  for the edit cost matrix:

$$\left( \begin{array}{cccc|c} 0 & 0 & 3 & \mathbf{0}^* & 4 \\ \mathbf{0}^* & 2 & 0 & 2 & 7 \\ 0 & \mathbf{0}^* & 2 & 3 & 6 \\ \hline 4 & 6 & 8 & 0 & 0 \end{array} \right), \quad G^0 =$$


The path starting with  $3 \in \mathcal{V}$  and ending with  $(n+1) \in \mathcal{U}_\epsilon$  leads to  $\rho=(1, 2, 3)$  and  $\varrho=(1, 2, 3, 4)$ :



For the last case, consider the partial  $\epsilon$ -assignment defined by  $\rho = (3, 4, 2)$  and  $\varrho = (0, 3, 1)$ . We want to assign  $1 \in \mathcal{V}$ :

$$\left( \begin{array}{ccc|c} 2 & 3 & \mathbf{0}^* & 4 \\ 7 & 0 & 5 & \mathbf{0}^* \\ 0 & \mathbf{0}^* & 4 & 6 \\ \hline 4 & 6 & 0 & 0 \end{array} \right), \quad G^0 = \begin{array}{c} \begin{array}{cccc} U_\epsilon & 1 & 2 & 3 & \epsilon \\ \bullet & \bullet & \bullet & \bullet & \circ \end{array} \\ \begin{array}{cccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ V_\epsilon & 1 & 2 & 3 & \epsilon \end{array} \end{array}$$

This leads to  $\rho = (3, 2, 1)$  and  $\varrho = (3, 2, 1)$ :

$$\begin{array}{c} \text{path:} \\ \begin{array}{cccc} U_\epsilon & 1 & 2 & 3 & \epsilon \\ \bullet & \bullet & \bullet & \bullet & \circ \end{array} \\ \begin{array}{cccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ V_\epsilon & 1 & 2 & 3 & \epsilon \end{array} \end{array} \xrightarrow{\text{swap}} \begin{array}{c} \begin{array}{cccc} U_\epsilon & 1 & 2 & 3 & \epsilon \\ \bullet & \bullet & \bullet & \bullet & \circ \end{array} \\ \begin{array}{cccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ V_\epsilon & 1 & 2 & 3 & \epsilon \end{array} \end{array} \Rightarrow \begin{array}{c} \begin{array}{cccc} U_\epsilon & 1 & 2 & 3 & \epsilon \\ \bullet & \bullet & \bullet & \bullet & \circ \end{array} \\ \begin{array}{cccc} \bullet & \bullet & \bullet & \bullet & \bullet \\ V_\epsilon & 1 & 2 & 3 & \epsilon \end{array} \end{array} \quad \left( \begin{array}{ccc|c} 2 & 3 & \mathbf{0}^* & 4 \\ 7 & \mathbf{0}^* & 5 & 0 \\ \mathbf{0}^* & 0 & 4 & 6 \\ \hline 4 & 6 & 0 & 0 \end{array} \right)$$

In this case, remark that the last edge of the path switch from assigned to unassigned. However, since columns corresponding to  $\epsilon$  are unconstrained, unassigning an element to  $\epsilon$  do not increase the number of elements to assign.

### 4.3 Construction of an augmenting path

Algorithm 2 details the construction of an augmenting path from an unassigned element  $k \in \mathcal{V}$ . It consists in growing a tree  $T^0$  of minimal alternating paths in  $G^0$ , rooted in  $k$ , until an augmenting path has been found. This is similar to Dijkstra's algorithm to compute minimal paths in bipartite graphs. The tree is constructed by growing two sets,  $SV$  (initialized to  $k$ ) and  $SU$ , representing the current nodes of the tree. Candidate nodes of  $\mathcal{U}$  to the extension are represented by the set  $LU \setminus SU$ , where the set  $LU$  defines the nodes of  $\mathcal{U}$  within the tree, as well as all candidate nodes. For computational purposes, the more general tree  $T$  of minimal alternating paths rooted in  $k$  (not necessary in  $G^0$ ) is constructed. This tree is encoded by a predecessor vector  $\mathbf{pred} \in \mathcal{V}^n$  (for unassigned edges) and the pair  $(\rho, \varrho)$  encoding the partial  $\epsilon$ -assignment (for assigned edges). The vector  $\pi \in [0, +\infty)^n$  encodes, for each  $i$ , the minimal transformed cost  $\bar{c}_{i,j}$  among all nodes  $j \in SV$ . The tree  $T^0 \subset G^0$  is obviously included in the tree  $T$ . Note that  $\pi$  and  $\mathbf{pred}$  do not consider the element  $\epsilon$ . Indeed, since it can only be a sink of an augmenting path, the algorithm stops when it is encountered (see below).

At each iteration, the last node  $j \in SV$  inserted to the tree  $T^0$  is considered (line 5). Each node  $i \in \mathcal{U} \setminus LU$  (line 9) is added to  $LU$  as a candidate node to the extension of  $T^0$  (line 15) if it is connected to  $j$  in  $G^0$  (line 13), *i.e.*  $\bar{c}_{i,j} = 0$ . Remark that  $T$  is constructed simultaneously (lines 10–12). When there is at least one candidate node ( $LU \setminus SU \neq \emptyset$ ), one is selected and added to the set  $SU$  of nodes of  $T^0$  (lines 30 and 31), *i.e.* the unassigned edge  $(i, \text{pred}_i)$  is candidate for an augmenting path, together with the assigned node  $\rho_i$  and edge  $(i, \rho_i)$ . Then  $\rho_i$  is considered for the next iteration (lines 32 and 5). The tree stops to grow when an unassigned node of  $\mathcal{U}$  is reached (line 14), when  $\epsilon$  is reached (line 14 or 6), or when there is no candidate node in  $LU$ , *i.e.*  $LU \setminus SU = \emptyset$  (line 16). In this case, dual variables  $\mathbf{u}$  and  $\mathbf{v}$  are updated such that at least one new unassigned edge is inserted in  $G^0$  (lines 18–22), *i.e.*  $\bar{c}_{i,j} = 0$  is satisfied for at least one pair  $(i, j) \in ((\mathcal{U} \setminus LU) \cup \{n+1\}) \times SV$ . Such nodes  $i \in \mathcal{U} \setminus LU$  become candidates (lines 26 and 28), as before. By construction,  $((\mathcal{U} \setminus LU) \cup \{n+1\}) \times SV$  does not contain any edge of  $G^0$  (including assigned edges).

**Proposition 6.** Let  $(\mathbf{X}, (\mathbf{u}, \mathbf{v}))$  and  $\mathbf{C}$  a partial  $\epsilon$ -assignment, a pair of dual variables and an edit cost matrix such that Eq. 32 is satisfied. Consider the minimum transformed cost

$$\delta = \min\{\bar{c}_{i,j} \mid i \notin LU, j \in SV\}$$

with  $\bar{\mathbf{C}} = \mathbf{C} - \mathbf{u}\mathbf{1}_{m+1}^T - \mathbf{1}_{n+1}\mathbf{v}^T$  and  $u_{n+1} = v_{m+1} = 0$ . Eq. 32 is still satisfied if the dual variables  $(\mathbf{u}, \mathbf{v})$  are updated by

$$\begin{cases} u_i \leftarrow u_i - \delta, & \forall i \in LU \\ v_j \leftarrow v_j + \delta, & \forall j \in SV \end{cases} \quad (38)$$

```

1 begin Augment( $k, \mathbf{C}, \rho, \varrho, \mathbf{u}, \mathbf{v}, U$ )
2    $n \leftarrow |U|, \pi \in [0, +\infty]^n, SV, SU, LU \leftarrow \emptyset, j \leftarrow k$ 
3   foreach  $i \in U$  do  $\pi_i \leftarrow +\infty$ 
4   while true do
5      $SV \leftarrow SV \cup \{j\}$ 
6     if  $(\varrho_j \leq n) \wedge (c_{n+1,j} - v_j = 0)$  then
7       return  $(n+1, j, \mathbf{u}, \mathbf{v}, \text{pred})$ 
8     // find candidate nodes and update the tree
9     foreach  $i \in U \setminus LU$  do
10      if  $c_{i,j} - u_i - v_j < \pi_i$  then
11         $\text{pred}_i \leftarrow j$ 
12         $\pi_i \leftarrow c_{i,j} - u_i - v_j$ 
13        if  $\pi_i = 0$  then
14          if  $\rho_i \in \{0, m+1\}$  return  $(i, 0, \mathbf{u}, \mathbf{v}, \text{pred})$ 
15           $LU \leftarrow LU \cup \{i\}$ 
16      // update dual variables and find candidate nodes
17      if  $LU \setminus SU = \emptyset$  then
18         $\delta_s \leftarrow \min\{\pi_i \mid i \in U \setminus LU\}$ 
19         $(l, \delta_\epsilon) \leftarrow (\text{argmin}, \min)\{c_{n+1,j} - v_j \mid j \in SV\}$ 
20         $\delta \leftarrow \min\{\delta_s, \delta_\epsilon\}$ 
21        foreach  $j \in SV$  do  $v_j \leftarrow v_j + \delta$ 
22        foreach  $i \in LU$  do  $u_i \leftarrow u_i - \delta$ 
23        if  $\delta_\epsilon \leq \delta_s$  return  $(n+1, l, \mathbf{u}, \mathbf{v}, \text{pred})$ 
24        foreach  $i \in U \setminus LU$  do
25           $\pi_i \leftarrow \pi_i - \delta$ 
26          if  $\pi_i = 0$  then
27            if  $\rho_i \in \{0, m+1\}$  return  $(i, 0, \mathbf{u}, \mathbf{v}, \text{pred})$ 
28             $LU \leftarrow LU \cup \{i\}$ 
29      // extend the tree
30       $i \leftarrow \text{any element in } LU \setminus SU$ 
31       $SU \leftarrow SU \cup \{i\}$ 
32       $j \leftarrow \rho_i$ 

```

**Algorithm 2:** Construct an augmenting path starting at  $k \in \mathcal{V}$ .

or equivalently if the transformed edit cost matrix  $\bar{\mathbf{C}}$  is updated by

$$\bar{c}_{i,j} \leftarrow \begin{cases} \bar{c}_{i,j} - \delta & \text{if } i \notin LU, j \in SV \\ \bar{c}_{i,j} + \delta & \text{if } i \in LU, j \notin SV \\ \bar{c}_{i,j} & \text{else} \end{cases} \quad (39)$$

This updating augments the graph  $G^0$  with at least one new edge in  $((U \setminus LU) \cup \{n+1\}) \times SV$ .

*Proof.* The updating of the dual variables given by Eq. 38 leads to the following cases:

1. When  $i \notin LU$  and  $j \notin SV$ , no updating occurs, and so the complementary slackness condition(CSC) is maintained in this case.
2. When  $i \in LU$  and  $j \in SV$ , we have  $c_{i,j} - (u_i - \delta) - (v_j + \delta) = c_{i,j} - u_i - v_j = \bar{c}_{i,j}$ . So no updating occurs and as before the CSC is preserved in this case.
3. When  $i \in LU$  and  $j \notin SV$ ,  $\bar{c}_{i,j}$  is updated by  $c_{i,j} - (u_i - \delta) - v_j = \bar{c}_{i,j} + \delta$ . So the transformed cost does not become negative. Since elements  $i \in LU$  can only be assigned to elements  $j \in SV$  ( $x_{i,j} = 1$ ), we have  $x_{i,j} = 0$  for all  $j \notin SV$ , and by consequence the CSC is preserved.
4. When  $i \notin LU$  and  $j \in SV$ ,  $\bar{c}_{i,j}$  is updated by  $c_{i,j} - u_i - (v_j + \delta) = \bar{c}_{i,j} - \delta$  which is non-negative by definition. Since  $\delta$  is the minimum of  $((U \setminus LU) \cup \{n+1\}) \times SV$ , the updating

produces at least one pair  $(i, j)$  such that  $\bar{c}_{i,j} = 0$  (a new edge added to  $G^0$ ). Also,  $\delta > 0$  induces that  $\bar{c}_{i,j} \neq 0$  and  $x_{i,j} = 0$  before the updating (by complementary slackness). By consequence the CSC is also preserved in this case.

So transformed costs do not become negative and the CSC is preserved on  $\mathcal{U}_\epsilon \times \mathcal{V}_\epsilon$ . Moreover, this shows that the graph  $G^0$  is augmented by at least one new edge, ensuring the dual updating to produce an enlarged tree.  $\square$

This shows that the dual update adds at least one edge to  $G^0$ , guaranteeing to complete the tree at each iteration (lines 30, 31 and 5). This also implies that Eq. 32 is satisfied at each iteration. Since each node of  $\mathcal{U}$  can be reached, and since each node of the tree  $T^0$  is inserted only once, an augmenting path is found if there is an unassigned node of  $\mathcal{U}$ . Since nodes  $\epsilon$  are unconstrained, an augmenting path is also found whenever they are encountered. Even when all elements of  $\mathcal{U}$  are already assigned, it is possible to find an augmenting path, ending by a node  $\epsilon$ . Since the length  $\gamma_{\bar{C}} = 0$ , by Proposition 5 the corresponding matrix  $\mathbf{X}'$  solves the sub-problem of the LSAP.

The growing of the tree depends on the encoding of the set  $LU \setminus SU$  and the selection of the next node of  $\mathcal{U}$  (line 30). We propose to use a FIFO strategy, leading to a breadth-first like growth of the tree. It can be efficiently encoded, together with the sets  $LU$ ,  $U \setminus LU$  and  $SU$ , by a permutation of  $\mathcal{U}$  (Fig. 6 and Appendix A), since  $\mathcal{U} = SU \cup (LU \setminus SU) \cup (U \setminus LU)$ . Remark that we can access to any element  $i \in \mathcal{U}$  in the array  $PU$  by  $PU(PU(i))$ , since  $PU$  is a permutation.

**Example 4.** Consider the partial  $\epsilon$ -assignment given by  $\rho = (3, 0, 1, 4, 0)$  and  $\varrho = (3, 6, 1, 4, 0, 0)$ , and the following cost matrix with  $\mathbf{u} = \mathbf{0}_5$  and  $\mathbf{v} = \mathbf{0}_6$ . We want to assign  $k = 5 \in \mathcal{V}$ , which is the root of the tree  $T$ . So in the first iteration of Algorithm 2, the element  $j = 5 \in \mathcal{V}$  is added to  $SV$  and all its neighbors in  $\mathcal{U}$  are added to the tree as a child of  $j$  ( $\pi$  and **pred**). Then, since there is one element  $i = 4 \in \mathcal{U}$  having a zero transformed cost  $\bar{c}_{i,j} = 0$ , we have  $LU \setminus SU = LU = \{4\}$ . Since there is one element  $j' = 4 \in \mathcal{V}$  such that  $\varrho_{j'} = i$ , it is selected as the next  $j$  for the next iteration and  $i = 4$  is then added to  $SU$ . The resulting tree  $T$  is given below, circles represent elements of  $\mathcal{V}$  and squares represent elements of  $\mathcal{U}$ . Black edges are used to represent  $T^0$ .

$$S/L \left( \begin{array}{cccc|cc|c} & & & S & & & \\ \hline 8 & 2 & \mathbf{0}^* & 1 & 2 & 1 & 3 \\ 2 & 4 & 4 & 6 & 5 & 7 & 1 \\ \mathbf{0}^* & 4 & 3 & 1 & 8 & 5 & 4 \\ 1 & 3 & 2 & \mathbf{0}^* & 0 & 0 & 1 \\ 2 & 0 & 1 & 3 & 4 & 5 & 3 \\ \hline 4 & \mathbf{0}^* & 1 & 7 & 8 & 4 & 0 \end{array} \right), \quad \pi = \begin{pmatrix} 2 \\ 5 \\ 8 \\ 0 \\ 4 \end{pmatrix}, \quad \mathbf{pred} = \begin{pmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{pmatrix}$$

In the second iteration, while there is no neighbor of  $j = 4$  in  $U \setminus LU$  with a zero reduced cost, the tree is updated if elements of  $U \setminus LU$  are connected to  $j$  by a strictly lower cost.

$$S/L \left( \begin{array}{cccc|cc|c} & & & S & S & & \\ \hline 8 & 2 & \mathbf{0}^* & 1 & 2 & 1 & 3 \\ 2 & 4 & 4 & 6 & 5 & 7 & 1 \\ \mathbf{0}^* & 4 & 3 & 1 & 8 & 5 & 4 \\ 1 & 3 & 2 & \mathbf{0}^* & 0 & 0 & 1 \\ 2 & 0 & 1 & 3 & 4 & 5 & 3 \\ \hline 4 & \mathbf{0}^* & 1 & 7 & 8 & 4 & 0 \end{array} \right), \quad \pi = \begin{pmatrix} 1 \\ 5 \\ 1 \\ 0 \\ 3 \end{pmatrix}, \quad \mathbf{pred} = \begin{pmatrix} 4 \\ 4 \\ 4 \\ 5 \\ 4 \end{pmatrix}$$

So a dual updating is performed since  $LU \setminus SU = \emptyset$ . The minimum cost is 1 and so the dual variables become  $\mathbf{u} = (0, 0, 0, -1, 0)$  and  $\mathbf{v} = (0, 0, 0, 1, 1, 0)$ .

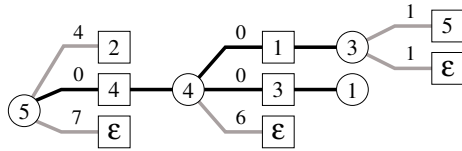
$$S/L \left( \begin{array}{cccc|cc|c} & & & S & S & & \\ \hline 8 & 2 & \mathbf{0}^* & \underline{1} & 2 & 1 & 3 \\ 2 & 4 & 4 & 6 & 5 & 7 & 1 \\ \mathbf{0}^* & 4 & 3 & \underline{1} & 8 & 5 & 4 \\ 1 & 3 & 2 & \mathbf{0}^* & 0 & 0 & 1 \\ 2 & 0 & 1 & 3 & 4 & 5 & 3 \\ \hline 4 & \mathbf{0}^* & 1 & 7 & 8 & 4 & 0 \end{array} \right), \quad \begin{cases} \delta_s = 1 \\ \delta_\epsilon = 7 \\ \delta = \min\{\delta_s, \delta_\epsilon\} = 1 \end{cases}, \quad \pi = \begin{pmatrix} 0 \\ 4 \\ 0 \\ 0 \\ 2 \end{pmatrix}, \quad \mathbf{pred} = \begin{pmatrix} 4 \\ 5 \\ 4 \\ 5 \\ 4 \end{pmatrix}$$

At the end of the iteration, we have the following edit cost matrix and tree, with  $LU \setminus SU = \{1, 3\}$ . By considering a FIFO strategy, the element  $i = 1$  is added to  $SU$  and its corresponding assigned element  $j = 3 \in \mathcal{V}$  is selected for the next iteration.

$$\begin{array}{c} L \\ L \\ S/L \end{array} \begin{array}{ccc|cc} 8 & 2 & \mathbf{0}^* & 0 & 1 \\ 2 & 4 & 4 & 5 & 4 \\ \mathbf{0}^* & 4 & 3 & 0 & 7 \\ \hline 2 & 4 & 3 & \mathbf{0}^* & 0 \\ 2 & 0 & 1 & 2 & 3 \\ 4 & \mathbf{0}^* & 1 & 6 & 7 \end{array} \begin{array}{c} S \\ S \end{array} \begin{array}{c} 1 \\ 3 \end{array} \begin{array}{c} | \\ | \\ | \\ | \\ | \\ | \end{array} \begin{array}{cc} 1 & 3 \\ 7 & 1 \\ 5 & 4 \\ 1 & 2 \\ 5 & 3 \\ 4 & 0 \end{array}, \quad \begin{array}{c} \text{Tree diagram showing nodes 5, 4, 3, 1, 2, 3, 1, 5, \epsilon with edges and weights.} \end{array} \quad i=1, j=3$$

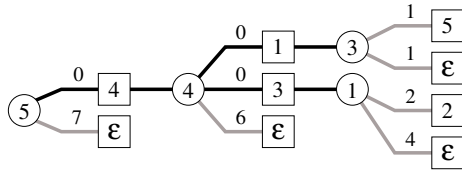
In the third iteration,  $j = 3 \in \mathcal{V}$  is added to  $SV$ , and the tree is updated. Since  $LU \setminus SU = \{3\}$ ,  $i = 3$  is added to  $SU$  and its corresponding element  $j = 1$  is selected for the next iteration.

$$\begin{array}{c} S/L \\ L \\ S/L \end{array} \begin{array}{ccc|ccc} 8 & 2 & \mathbf{0}^* & 0 & 1 & 1 \\ 2 & 4 & 4 & 5 & 4 & 7 \\ \mathbf{0}^* & 4 & 3 & 0 & 7 & 5 \\ \hline 2 & 4 & 3 & \mathbf{0}^* & 0 & 1 \\ 2 & 0 & 1 & 2 & 3 & 5 \\ 4 & \mathbf{0}^* & 1 & 6 & 7 & 4 \end{array} \begin{array}{c} S \\ S \\ S \end{array} \begin{array}{c} 3 \\ 3 \\ 1 \end{array} \begin{array}{c} | \\ | \\ | \\ | \\ | \\ | \end{array} \begin{array}{ccc} 3 \\ 1 \\ 1 \\ 5 \\ 3 \\ 0 \end{array}, \quad \pi = \begin{pmatrix} 0 \\ 4 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \mathit{pred} = \begin{pmatrix} 4 \\ 5 \\ 4 \\ 5 \\ 3 \end{pmatrix}$$



In the fourth iteration,  $j = 1 \in \mathcal{V}$  is added to  $SV$ .

$$\begin{array}{c} S/L \\ S/L \\ S/L \end{array} \begin{array}{ccc|ccc} 8 & 2 & \mathbf{0}^* & 0 & 1 & 1 \\ 2 & 4 & 4 & 5 & 4 & 7 \\ \mathbf{0}^* & 4 & 3 & 0 & 7 & 5 \\ \hline 2 & 4 & 3 & \mathbf{0}^* & 0 & 1 \\ 2 & 0 & 1 & 2 & 3 & 5 \\ 4 & \mathbf{0}^* & 1 & 6 & 7 & 4 \end{array} \begin{array}{c} S \\ S \\ S \\ S \end{array} \begin{array}{c} 3 \\ 3 \\ 1 \\ 1 \end{array} \begin{array}{c} | \\ | \\ | \\ | \\ | \\ | \end{array} \begin{array}{ccc} 3 \\ 1 \\ 1 \\ 5 \\ 3 \\ 0 \end{array}, \quad \pi = \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \mathit{pred} = \begin{pmatrix} 4 \\ 1 \\ 4 \\ 5 \\ 3 \end{pmatrix}$$



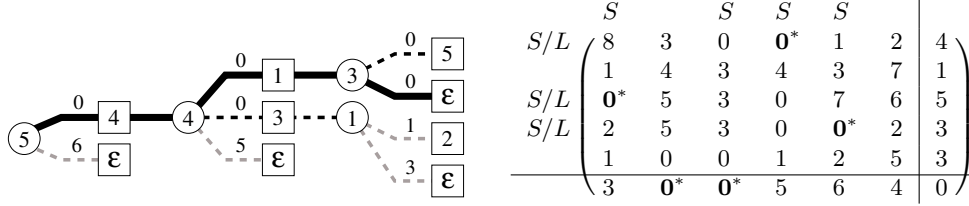
Since  $LU \setminus SU = \emptyset$ , dual variables are updated. The minimum is  $\delta = 1$  and the dual variables are updated as before:  $\mathbf{u} = (-1, 0, -1, -2, 0)$  and  $\mathbf{v} = (1, 0, 1, 2, 2, 0)$ . According to

$$\begin{array}{c} S/L \\ S/L \\ S/L \end{array} \begin{array}{ccc|ccc} 8 & 2 & \mathbf{0}^* & 0 & \underline{1} \\ 2 & 4 & 4 & 5 & 4 \\ \mathbf{0}^* & 4 & 3 & 0 & 7 \\ \hline 2 & 4 & 3 & \mathbf{0}^* & 0 \\ 2 & 0 & \underline{1} & 2 & 3 \\ 4 & \mathbf{0}^* & \underline{1} & 6 & 7 \end{array} \begin{array}{c} S \\ S \\ S \\ S \end{array} \begin{array}{c} 3 \\ 3 \\ 1 \\ 1 \end{array} \begin{array}{c} | \\ | \\ | \\ | \\ | \\ | \end{array} \begin{array}{ccc} 3 \\ 1 \\ 1 \\ 5 \\ 3 \\ 0 \end{array}, \quad \pi = \begin{pmatrix} 0 \\ 2 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \quad \mathit{pred} = \begin{pmatrix} 4 \\ 1 \\ 4 \\ 5 \\ 3 \end{pmatrix}$$

the updating of the transformed costs is given by

$$\begin{array}{c} S \\ S/L \end{array} \begin{array}{c} S \\ S \\ S/L \\ S/L \end{array} \begin{array}{c} S \\ S \\ S \\ S \end{array} \left( \begin{array}{cccc|c} 8 & 2 & \mathbf{0}^* & 0 & 1 & 1 & 3 \\ 1 & 4 & 3 & 4 & 3 & 7 & 1 \\ \mathbf{0}^* & 4 & 3 & 0 & 7 & 5 & 4 \\ 2 & 4 & 3 & \mathbf{0}^* & 0 & 1 & 2 \\ 1 & 0 & 0 & 1 & 2 & 5 & 3 \\ \hline 3 & \mathbf{0}^* & 0 & 5 & 6 & 4 & 0 \end{array} \right), \quad \pi = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad \mathbf{pred} = \begin{pmatrix} 4 \\ 1 \\ 4 \\ 5 \\ 3 \end{pmatrix}$$

Remark that there is two minimal paths. But, according to Algorithm 2,  $\epsilon \in \mathcal{U}_\epsilon$  is selected as a sink, and an augmenting path can then be extracted as described in the following section.



$$\begin{array}{c} S \\ S/L \\ S/L \\ S/L \end{array} \begin{array}{c} S \\ S \\ S \\ S \end{array} \left( \begin{array}{cccc|c} 8 & 3 & 0 & \mathbf{0}^* & 1 & 2 & 4 \\ 1 & 4 & 3 & 4 & 3 & 7 & 1 \\ \mathbf{0}^* & 5 & 3 & 0 & 7 & 6 & 5 \\ 2 & 5 & 3 & 0 & \mathbf{0}^* & 2 & 3 \\ 1 & 0 & 0 & 1 & 2 & 5 & 3 \\ \hline 3 & \mathbf{0}^* & \mathbf{0}^* & 5 & 6 & 4 & 0 \end{array} \right)$$

**Proposition 7.** Algorithm 2 computes an augmenting path in  $O((2n + \min\{n, m\}) \min\{n, m\})$  time complexity.

*Proof.* At each iteration of the main loop, one element of  $\mathcal{V}$  is added to  $SV$  (initially empty). For the first iteration this element is provided ( $j=k$  with  $k$  being the root of the tree), but for all other iterations it is computed from the last element  $i \in \mathcal{U}$  added to  $SU$  (also initially empty). By assuming that a sink is always found by the algorithm, we have  $|SU| = |SV| - 1$  since the main loop ends before augmenting  $SU$ . Moreover, since elements are added only once to  $SU$  and  $SV$ , we have  $|SU| \leq |\mathcal{U}| - 1$  and  $|SV| \leq |\mathcal{V}|$ . Then we have  $|SV| = |SU| + 1 \leq |\mathcal{U}|$ , and by consequence  $|SV| \leq \min\{|\mathcal{U}|, |\mathcal{V}|\}$  and  $|SU| \leq \min\{|\mathcal{U}|, |\mathcal{V}|\} - 1$ . So the main loop has  $O(\min\{n, m\})$  iterations. For each iteration:

- $\mathcal{U} \setminus LU$  is traversed 3 times (lines 9, 17 and 24) and has at most  $n$  elements,
- $LU$  is traversed 1 time (line 22) and has at most  $n$  elements,
- $SV$  is traversed 2 times (lines 19 and 21) and has at most  $\min\{n, m\}$  elements.

So each iteration of the main loop runs in  $O(2n + \min\{n, m\})$  time complexity. Indeed, the encoding of the sets  $LU$ ,  $LU \setminus SU$  and  $\mathcal{U} \setminus LU$  with the permutation  $PU$  allows to traverse these sets in  $O(n)$  time complexity.  $\square$

Observe that the time complexity depends on  $\min\{n, m\}$ . If  $n \leq m$  then Algorithm 2 runs in  $O(n^2)$ , else in  $O((2n + m)m)$ .

Algorithm 2 assigns an unassigned node  $k \in \mathcal{V}$ . An unassigned node  $k \in \mathcal{U}$  can also be assigned by simply swapping the role for  $\rho$  and  $\varrho$ ,  $\mathbf{u}$  and  $\mathbf{v}$ , and by considering  $\mathbf{C}^T$ .

#### 4.4 Main algorithm

Let  $(\rho, \varrho)$  and  $(\mathbf{u}, \mathbf{v})$  be a partial  $\epsilon$ -assignment and its associated dual variables satisfying Eq. 32, obtained for instance with Algorithm 1.

The partial  $\epsilon$ -assignment is completed by Algorithm 3. To this, each unassigned element of  $\mathcal{V}$  is assigned to an element of  $\mathcal{U}_\epsilon$  by computing an augmenting path (Algorithm 2) and then by swapping unassigned and assigned edges along the path (2nd step of assignCols and Proposition 5), which is realized by a backtrack from the sink to the root of the tree according to the predecessor vector and the previous partial  $\epsilon$ -assignment in  $O(\min\{n, m\})$  time complexity. Since  $m$  elements of  $\mathcal{V}$  can be unassigned, the first call to assignCols is executed in  $O(m \min\{n, m\}(2n + \min\{n, m\})) = O(\min\{n, m\}^2(m + 2 \max\{n, m\}))$  time complexity.



```

1 begin HungarianLSAPE( $\mathbf{C}, \rho, \varrho, \mathbf{u}, \mathbf{v}$ )
2    $U \leftarrow \{1, \dots, n\}, V \leftarrow \{1, \dots, m\}$ 
3   // assign unassigned elements of  $V$ 
4    $(\rho, \varrho, \mathbf{u}, \mathbf{v}) \leftarrow \text{assignCols}(\mathbf{C}, \rho, \varrho, \mathbf{u}, \mathbf{v}, U, V)$ 
5   // assign unassigned elements of  $U$ 
6    $(\varrho, \rho, \mathbf{v}, \mathbf{u}) \leftarrow \text{assignCols}(\mathbf{C}^T, \varrho, \rho, \mathbf{v}, \mathbf{u}, V, U)$ 
7   return  $((\rho, \varrho), (\mathbf{u}, \mathbf{v}))$ 

8 begin assignCols( $\mathbf{C}, \rho, \varrho, \mathbf{u}, \mathbf{v}, U, V$ )
9   foreach  $k \in V \mid \varrho_k = 0$  do
10    // find an augmenting path rooted in  $k$ 
11     $(i, j, \mathbf{u}, \mathbf{v}, \mathbf{pred}) \leftarrow \text{Augment}(k, \mathbf{C}, \rho, \varrho, \mathbf{u}, \mathbf{v}, U)$ 
12    // update partial primal solution
13    if  $i = n + 1$  then  $r \leftarrow \varrho_j, \varrho_j \leftarrow i, i \leftarrow r$ 
14    else  $j \leftarrow 0$ 
15    while  $j \neq k$  do
16       $j \leftarrow \text{pred}_i, \rho_i \leftarrow j$ 
17       $r \leftarrow \varrho_j, \varrho_j \leftarrow i, i \leftarrow r$ 
18  return  $(\rho, \varrho, \mathbf{u}, \mathbf{v})$ 

```

**Algorithm 3:** Compute an  $\epsilon$ -assignment.

The second step of Algorithm 3 consists in augmenting similarly the remaining unassigned elements of  $\mathcal{U}$  (line 6). This case occurs when  $m < n$ , or when more than  $\max\{n, m\} - \min\{n, m\}$  elements are assigned to  $\epsilon$  in the first step. In the worst-case, all elements of  $\mathcal{V}$  have been assigned to  $\epsilon$  in the first step and no element of  $\mathcal{U}$  have yet been assigned. So the second step has  $O(\min\{n, m\}^2(n + 2 \max\{n, m\}))$  time complexity.

**Proposition 8.** *Algorithm 3 solves the LSAPÉ and its dual problem in  $O(\min\{n, m\}^2 \max\{n, m\})$  time complexity and in  $O(nm)$  space complexity.*

*Proof.* At each iteration of Algorithm 2, the complementary slackness condition is satisfied. So Algorithm 3 computes a maximal set  $S = \{(i, \rho_i) \mid i \in \mathcal{V}\} \cup \{(\varrho_j, j) \mid j \in \mathcal{V}, \varrho_j = n + 1\}$  of  $\epsilon$ -independent elements satisfying  $c_{i,j} = u_i + v_j$  for all  $(i, j) \in S$  and  $c_{i,j} \geq u_i + v_j$  for all  $(i, j) \notin S$ . Then by Corollary 3 the  $\epsilon$ -assignment is optimal. From the above discussion on complexities, we have

$$\begin{aligned}
& \min\{n, m\}^2(m + 2 \max\{n, m\}) + \min\{n, m\}^2(n + 2 \max\{n, m\}) \\
&= \min\{n, m\}^2(n + m + 4 \max\{n, m\}) \\
&= \min\{n, m\}^2(\min\{n, m\} + 5 \max\{n, m\}) \\
&\leq 6 \min\{n, m\}^2 \max\{n, m\}
\end{aligned}$$

which completes the proof.  $\square$

Assume that  $n \leq m$ , the LSAPÉ is thus solved in  $O(n^2m)$  time complexity by Algorithm 3. Recall that the LSAPÉ is equivalent the sLSAPÉ (Problem 2). The complexities of the proposed algorithm (Proposition 8) are lower than the ones obtained for solving the sLSAPÉ with the Hungarian algorithm, *i.e.*  $O((n+m)^3)$  in time and  $O((n+m)^2)$  in space. Observe that more  $n, m$ , and  $|m - n|$  are important, more the improvement is.

This is confirmed in practice. We have compared the execution time of the LSAPÉ and the sLSAPÉ with three types of synthetic edit cost matrices:

1. random matrices (Fig. 18),
2. matrices of general term  $c_{i,j} = i * j$  (Fig. 18, 1st row),
3. matrices of general term  $c_{n-i+1, m-j+1} = i * j$  for  $i \in \{1, \dots, n\}$  et  $j \in \{1, \dots, m\}$ . Last line and column are respectively defined as copies of the  $n - th$  line and the  $m - th$  column (Fig. 18, 2nd row).

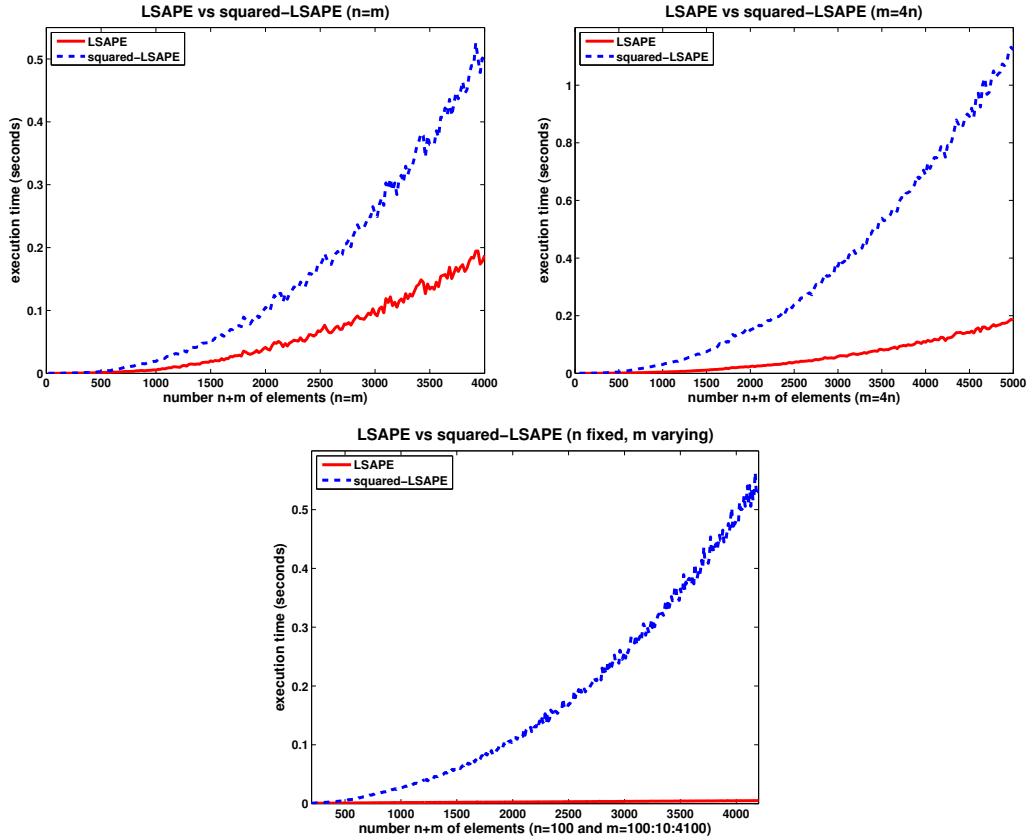


Figure 4: Execution time (in seconds) of the LSAP (red) vs. squared-LSAP (blue dotted) for (uniform) random edit cost matrix. For each size  $n + m$  ( $x$ -abscissa), a new edit cost matrix is generated and the time is averaged over several executions on this matrix. First row: both  $n$  and  $m$  are varying. Second row:  $n$  is fixed and  $m$  is varying.

The sLSAP is solved by an algorithm similar to Algorithm 3 (see [4]) wherein only one set needs to be augmented and all instructions related to  $\epsilon$  are not considered. While the LSAP and the sLSAP provide the same optimal value  $A_\epsilon$ , as expected the proposed algorithm runs faster. The difference in execution time between both algorithms increases according to the difference  $|m - n|$ . This is easily explained by the growth of the number of assignments between  $\epsilon$ -elements in the sLSAP.

The C++ source code used for these experiments is freely downloadable<sup>1</sup>.

## Conclusion

We have presented a general framework to transform a set into another by means of edit operations (substitutions, removals and insertions). The problem of finding a minimal transformation is formalized as a linear sum assignment problem where elements can be removed and inserted. This problem was previously solved by augmenting the given sets such that it can be formalized as a classical linear sum assignment problem, and thus solved by well-known methods such as the Hungarian algorithm. Based on the proposed model, the problem is solved by an adaptation of the Hungarian algorithm that has lower time and memory complexities. Based on the same model, other algorithms can be adapted similarly.

<sup>1</sup><http://forge.greyc.fr/projects/lshape>

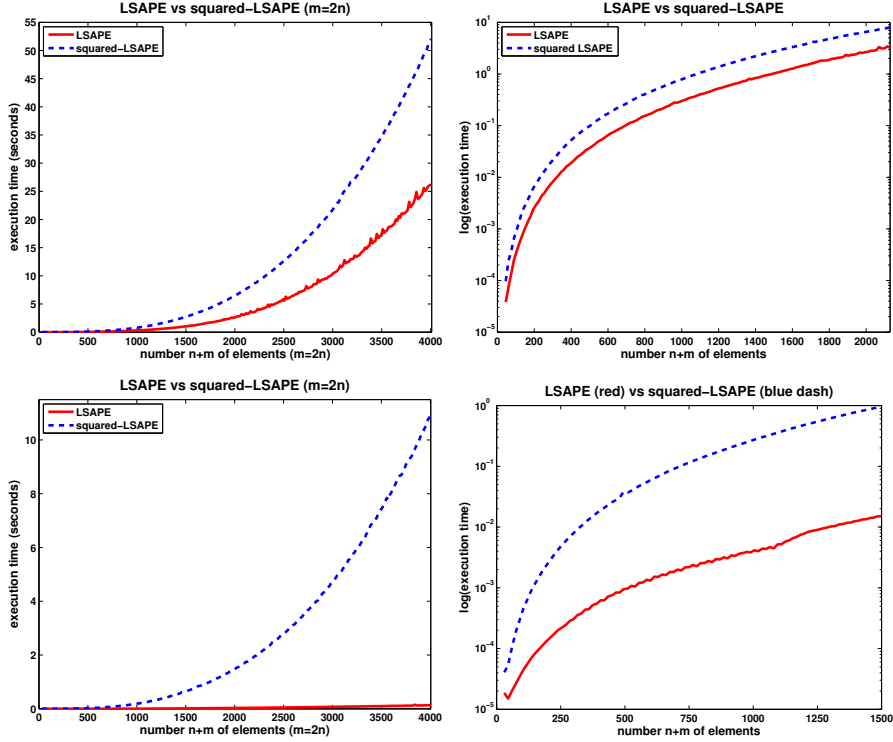


Figure 5: Left column: execution time (in seconds) of the LSAP (red) vs. squared-LSAP (blue dotted). Right column: associated log plot. First row: worst-case obtained for  $c_{i,j} = ij$ . Second row: a variant showing an important improvement (see text).

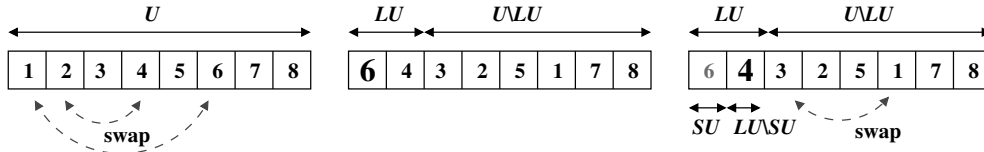


Figure 6: Encoding of the sets involved in Algorithm 2 as a permutation  $PU$  of  $U$ .

## A Managing the sets in Algorithm 2

Recall that the sets  $U \setminus LU$ ,  $LU$ ,  $SU$  and  $LU \setminus SU$  are represented by a permutation  $PU$  of  $U$  (Fig. 6). A link to the beginning of each set in the permutation is also saved.

**Example 5.** Consider a set  $U = \{1, \dots, 8\}$ . At the beginning of Algorithm 2, we have  $PU = U$  (Fig. 6 left),  $LU = SU = \emptyset$ , and so  $LU \setminus SU = \emptyset$ . Assume that  $\pi_4 = \pi_6 = 0$  in the first step, and so  $i = 4, 6$  are iteratively added to  $LU$  by swapping each of them with the top of  $U \setminus LU$  (initially equal to the top of  $PU$ ), and by incrementing the link to the top. Then we have  $LU = \{4, 6\}$ , the link to the top of  $U \setminus LU$  is the third element of the array, and the link to the top of  $LU \setminus SU$  is the first element of the array (Fig. 6 middle). There is no dual update since  $LU \setminus SU \neq \emptyset$ , so the first iteration of the main loop of Algorithm 2 ends by selecting the top of  $LU \setminus SU$ , i.e.  $i = 6$ . Then we have  $SU = \{6\}$ ,  $LU \setminus SU = \{4\}$ , and the link to the top of  $LU \setminus SU$  is incremented in consequence to become the second element of the array (Fig. 6 right).

Now assume that in the second iteration of the main loop, in the first step, we have  $\pi_1 = 0$ . The element  $i = 1$  is added to  $LU$  by swapping it with the top of  $U \setminus LU$ , and the process is going on as before until a sink is found.

## References

- [1] F. Bourgeois and J.C. Lassalle. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Commun. ACM*, 14:802–804, 1971.
- [2] H Bunke. Error correcting graph matching: on the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):917–922, 1999.
- [3] H Bunke and G Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters*, 1(4):245–253, 1983.
- [4] R. Burkard, M. Dell’Amico, and S. Martello. *Assignment Problems*. SIAM, 2009.
- [5] S. Fankhauser, K. Riesen, and H. Bunke. Speeding up graph edit distance computation through fast bipartite matching. In *Graph-Based Representations in Pattern Recognition*, volume 6658 of *LNCS*, pages 102–111. Springer Berlin Heidelberg, 2011.
- [6] B. Gaüzère, S. Bogleux, K. Riesen, and L. Brun. Approximate graph edit distance guided by bipartite matching of bags of walks. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume 8621 of *LNCS*, pages 73–82, 2014.
- [7] D. Justice and A. Hero. A binary linear programming formulation of the graph edit distance. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(8):1200–1214, 2006.
- [8] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.
- [9] H.W. Kuhn. Variants of the hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 3:253–258, 1956.
- [10] E.L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York, 1976.
- [11] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.
- [12] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27:950–959, 2009.
- [13] K. Riesen and H. Bunke. Improving bipartite graph edit distance approximation using various search strategies. *Pattern Recognition*, 28(4):1349–1363, 2015.
- [14] K. Riesen, X. Jiang, and H. Bunke. Exact and inexact graph matching: Methodology and applications. In C.C. Aggarwal and H. Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 217–247. Springer US, 2010.
- [15] K. Riesen, M. Neuhaus, and H. Bunke. Bipartite graph matching for computing the edit distance of graphs. In *Graph-Based Representations in Pattern Recognition*, volume 4538 of *LNCS*, pages 1–12. Springer Berlin Heidelberg, 2007.
- [16] A. Sanfeliu and K.-S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):353–362, 1983.
- [17] S. Serratos. Speeding up fast bipartite graph matching through a new cost matrix. *Int. Journal of Pattern Recognition*, 29(2), 2015.
- [18] G. Sierksma and Y. Zwols. *Linear and Integer Optimization: Theory and Practice*. Advances in Applied Mathematics. Chapman and Hall/CRC, 3rd edition, 2015.
- [19] W.-H. Tsai and K.-S. Fu. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *IEEE Trans. on Systems, Man and Cybernetics*, 9(12):757–768, 1979.
- [20] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou. Comparing stars: On approximating graph edit distance. *Proceedings of the VLDB Endowment*, 2(1):25–36, 2009.