The Damerau-Levenshstein distance function is a modified version of the Levenshstein distance[3], which is a widely used to compute the edit-distance of two discrete sequences[1, 4]. The most important applications are within the Natural Language Processing (NLP) discipline and the Biomedical Sciences. Within these areas, we often use the Levenshtein distance to compute the edit-distance between two words, two sentences or two DNA sequences. Note, that the elements of these sequences are often atomic symbols instead of multidimensional vectors. Generally, the distance accounts for inserts, deletions and substitutions of elements between the two serquences. Damerau modified the distance function to allow for transposition operations. For Process Mining, transpositions are important as one event can transition into two events that are processed in parallel and may have varying processing times[2]. In Figure 1, we schematically show two sequences and their distance.

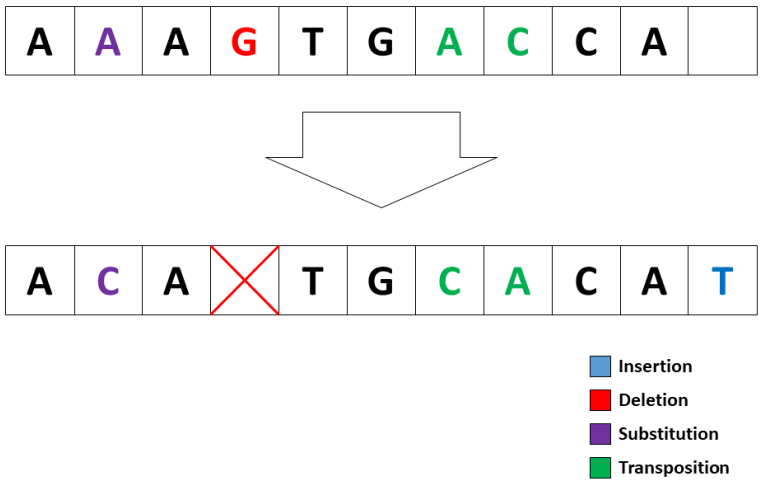

Figure 1: Two arbitrary sequences and their edit difference according to Damerau. The edit distance is the sum of each operation necessary to transform the sequence to another sequence. Blue shows an insert, red a deletion, purple a substitution and green a transposition. Therefore, the edit distance is 4.

Equation 1 depicts the recursive formulation of the distance. The distance computes the costs of transforming the sequence a to b, by computing the

minimum of five seperate terms.

$$d_{a,b}(i,j) = \min \begin{cases} d_{a,b}(i-1,j)+1 & \text{if } i > 0 \\ d_{a,b}(i,j-1)+1 & \text{if } j > 0 \\ d_{a,b}(i-1,j-1)+1 & \text{if } i,j > 0 \\ d_{a,b}(i-2,j-2)+1 & \text{if } i,j > 1 \wedge a_i = b_{j-1} \wedge a_{i-1} = b_j \\ 0 & \text{if } i = j = 0 \end{cases} \tag{1}$$

The recursive form $d_{a,b}(i,j)$ for sequences a and b with respective elements i and j takes the minimum of each allowed edit operation. In particular, no change, deletion, insertion, substitution and transposition. For each operation, the algorithm adds an edit cost of 1.

We cannot use the Damerau-Levenshstein distance for process mining if the process carries additional information about event attributes. Mainly, because two events may be emitted by the same activity, but they may still carry different event attributes.

To illustrate the issue, we explore a couple of examples. Let us assume, we have two strings $s^1 = aaba$ and $s^2 = acba$. Using the Damerau-Levenshstein distance, the edit distance between both sequences is 1, as we can recognise a substitution at the second position in both strings. However, this representation is insufficient for process instances. Therefore, we now characterise the two sequences as process events rather than strings in Equation 2.

$$s^1 = \{a, a, b, a\} \tag{2}$$
$$s^2 = \{a, a^*, b, a\} \tag{3}$$
$$s^3 = \{a, c, b, a\} \tag{4}$$
$$s^4 = \{a, a, b\} \qquad a, b, c \in \mathbb{R}^3 \tag{5}$$
$$a = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix} \; a^* = \begin{bmatrix} 3 \\ 3 \\ 4 \end{bmatrix} \; b = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \; c = \begin{bmatrix} 3 \\ 1 \\ 4 \end{bmatrix} \tag{6}$$

If we do not consider attribute values, it becomes clear that s^2, s^3 and s^4 have an edit-distance to s^1 of 0, 1 and 1. However, with attribute values in mind, s^1 and s^2 display clear differences. Similarly, s^1 and s^3 not only differ in terms of activity but also attribute value. Lastly, s^1 and s^4 are the same in attribute values, but one element still misses entirely. It appears unintuitive that each of these differences are associated with the same cost. The examples show that we can neither disregard attribute values nor events, while computing the edit distance of two process instances.

Instead, we have to define a cost function which takes attribute variables into account. Therefore, we modify the Damerau-Levenshstein distance by introducing a cost function instead of a static cost. Here, the cost of each edit-type is determined by a distance-function, which considers the difference between event-attributes. Therefore, we propose an edit-function, which captures structural sequence differences, as well as, content related differences. Going back to our example, if assume our cost function to only count differences in attributes, then the difference between s^1 and s^2 shall be 2 as their activities are the same, but the first two event attributes are different. To illustrate the structural elements, the difference between s^1 and s^3 shall be 3 instead of 2. Even if both a and c have two common event attributes, the activities they represent are still different. For instance, if both s^1 and s^3 were medical processes and a and c represented taking a cancer drug or a placebo, anyone would understand both activities are different even if the patient took the same dosage.