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. In Figure 1, we schematically show two sequences and their distance.

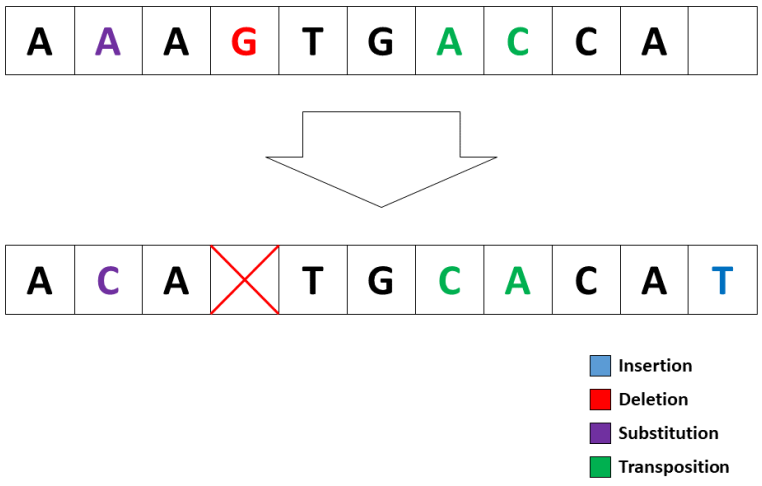

Figure 1: Shows two sequences. The edit distance is the sum of multiple operations. Blue shows an insert, red a deletion, purple a substitution and green a transposition. Therefore the edit distance is 4.

In 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 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. For Process Mining, it becomes necessary to modify the distance further.

To illustrate the issue, we explore a couple of examples. Lets assume, we have two strings $s^1 = aaba$ and $s^2 = acba$. Using the Damerau-Levenshtein-Distance, the edit distance between both sequences is zero, as we can recognise one substitution at the second character in both strings. However, this representation is insufficient for process instances as they may also contain attibute values. Therefore, we characterise the sequences as process events in Equation 2.

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

If we do not consider attribute values, it becomes clear that s^2, s^3 and s^4 have an edit distance of 0, 1 and 1 to s^1. However, with attribute values 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. These examples show that we can neither disregard attribute values nor events, while computing the edit distance of two process instances. We show this in **??**. In other words, we cannot simply assume a static cost of 1 for each edit operation. Instead, we have to define a cost function which takes attribute variables

into account. In the following sections, we will establish distances which use a modified Damerau-Levenshstein distance approach. Here, the cost of each edit-operation will be weighted with a distance function that considers the difference between event attributes. In simplified terms, we can say that s^1 and s^2 are identical, if we only consider the activity. However, taking attribute values into account, s^1 and s^2 actually differ on two accounts.
In order to reflect these differences in attribute values, we introduce a modified version of the Damerau-Levenshstein distance, that not only reflects the difference between two process instances, but also the attribute values. We achieve this by introducing a cost function $cost_{a_i,b_j}$, which applies to a normed vector-space[1]. Concretely, we formulate the modified Damerau-Levenshstein distance as shown in Equation 7. For the remainder, we will denote this edit-distance as Semi-strucured Damerau-Levenshtein distance (SSDLD).

$$d_{a,b}(i,j) = \min \begin{cases} d_{a,b}(i-1,j) + cost(\mathbf{0}, b_j) & \text{if } i > 0 \\ d_{a,b}(i,j-1) + cost(a_i, \mathbf{0}) & \text{if } j > 0 \\ d_{a,b}(i-1,j-1) + cost(a_i, b_j) & \text{if } i,j > 0 \\ & \quad \& \ \overline{a}_i = \overline{b}_j \\ d_{a,b}(i-1,j-1) + cost(a_i, \mathbf{0}) + cost(\mathbf{0}, b_j) & \text{if } i,j > 0 \\ & \quad \& \ \overline{a}_i \neq \overline{b}_j \\ d_{a,b}(i-2,j-2) + cost(a_i, b_{j-1}) + cost(a_{i-1}, b_j) & \text{if } i,j > 1 \\ & \quad \& \ \overline{a}_i = \overline{b}_{j-1} \\ & \quad \& \ \overline{a}_{i-1} = \overline{b}_j \\ 0 & \quad \& \ i = j = 0 \end{cases} \tag{7}$$

Here, $d_{a,b}(i,j)$ is the recursive form of the Damerau-Levenshtein-Distance. a and b are sequences and i and j specific elements of the sequence. $cost(a,b)$ is a cost function which takes the attribute values of a and b into account. The first two terms correspond to a deletion and an insertion from a to b. The idea is to compute the maximal cost for that the wrongfully deleted or inserted event. The third term adds the difference between two events with identical activities $\overline{a}_i$ and $\overline{b}_j$. As mentioned earlier, two events that refer to the same activity can still be different due to event attributes. The distance between the event attributes determines *how* different these events are. The fourth term handles the substitution of two events. Here, we compute the

[1] A normed vector-space is a vector space, in which all vectors have the same dimensionality. For instance, if all vectors have three dimensions, we can call the vector-space *normed*.

substitution cost as the sum of an insertion and a deletion. The fifth term computes the cost after transposing both events. This cost is similar to term 3 only that we now consider the differences between both events after they were aligned. The last term relates to the stopping criterion of the recursive formulation of the Damerau-Levenshstein distance.