MASTER

A generic similarity metric for activity sequences
with an application in Truck Routing Schedule search

de Vries, M.P.

Award date:
2017

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
A Generic Similarity Metric for Activity Sequences
With an Application in Truck Routing Schedule Search

Michiel de Vries MSc
August 23, 2017

Master’s thesis
Student identity number: 0748767
in partial fulfillment of the requirements for the degree of
Master of Science in Operations Management and Logistics

Supervisors:
Dr.ir. Remco Dijkman, Eindhoven University of Technology
Dr.ir. Willem Jaarsveld, Eindhoven University of Technology
Dr.ir. Stijn-Pieter van Houten, Accenture Strategy
Drs. Hans Peter van Dam, Accenture Strategy
TUE. School of Industrial Engineering
Series Master Theses Operations Management and Logistics

Subject headings: similarity measurement, activity sequence comparison, truck route comparison, similarity-based decision support (SDSS)
Abstract

The comparison of activity sequences often requires detailed knowledge about the rationale behind the order of activities in a sequence. An activity sequence is defined as a series of sequential processes that are individually characterized by a start time, a duration and a process classification. Few studies have examined mechanisms to compare the structure of activity sequences without prior knowledge of their construction logic. Disregarding this prior knowledge in a comparison leads to a descriptive assessment, with a possible identification of implicitly made construction decisions. Moreover, structural comparison of activity sequences can be used for similarity determination between activity sequences. Structural comparison and similarity determination can be used for searches and structural difference clarification.

To fill this research gap, a generic metric to structurally compare activity sequences is proposed. The metric is composed of similarity score factors resulting from two measurement steps. The first step involves uncovering the most efficient edit operations to obtain activity sequence $y$ from sequence $x$. These edit operations are found by back-tracing edit costs allocated by the traditional Levenshtein and Damerau-Levenshtein text similarity metrics. The second step involves the comparison of corresponding activities in terms of duration and start time. Subsequently, the similarity score factors are trained and parametrized using expert input in the context where similarity comparison and structural difference clarification is required.

In the context of truck routing schedule comparison, the proposed metric showed 23.5% worse and 9.5% better than random accuracy for the Levenshtein and Damerau-Levenshtein based metric respectively ($n=525$). Moreover, the Levenshtein based metric accuracy increases together with the number of corresponding activities. The overall achieved accuracy in this context can be considered poor.

Finally, this thesis provides a metric to measure similarity among activity sequences in any context based on their structural properties. Last, application areas in decision support and solution generation are described.
Preface

This Master’s thesis demonstrates the result of a six month graduate research project, in partial fulfillment of the degree of Master of Science in Operations Management and Logistics at the faculty Industrial Engineering and Innovation Sciences.

I have had the opportunity to work independently for the greater part of this thesis with a panel of supervisors on stand-by that would reply to an email within the hour. I would hereby like to thank my supervisors Remco, Willem, Stijn-Pieter and Hans Peter for their involvement in this research.

Remco, I find it a privilege to have worked under your supervision. I have enjoyed our meetings partly because of their high productivity and a great bit because all meetings overran due to talking about sports, side projects and life as a scholar. Besides, if it was not for you, I would not have learned that theoretical or conceptual research can be inspiring, or even fun.

Stijn-Pieter, your business orientation and critical but stimulating attitude have motivated me to challenge myself to keep pushing the bar of this research. I want to thank you for your involvement and approachability. I have enjoyed working with you.

Hans Peter, as the alpha-guy of this group, you have kept an eye on business relevance and understandability of this not-too-alpha piece of work. Your down-to-earth view on the overall progress has kept me from losing myself in exploring side paths.

Accenture Strategy has been a fantastic environment for me to do my graduation project. Upon starting, I was embraced as part of the team. Colleagues were soon seen as friends with motivating words and not only taking time for support and critical feedback but also to show genuine interest in this scientific, slightly technical research project. In the last six months I have experienced an unmatched dedication and enthusiasm towards project work that has impressed, attracted and motivated me.

Finally, in the past 18 months I have learned more than in any period of my life. Recovering from a severe concussion has put my life upside down and made me re-evaluate the true virtues of life, the value of one’s brain and the appreciation towards inspiring surrounding people.

The presented work is the fruit of learning, prototyping and perseverance. I sincerely hope that you will enjoy reading it.

Michiel de Vries
"If you can’t explain it simply, you don’t understand it well enough.”

-Albert Einstein-
Executive Summary

This research presents a method to compare sequences of activities in a descriptive, generic way. The comparison is made based on the structural properties of the assessed activity sequences. This method can be used for processes such as (local) search and sequence improvement applications. These processes can function as an activity sequence search engine, improve problem solving speed and quality, generate improvement suggestions and automate sequence invention.

An activity sequence is defined as a set of activities, or time-consuming tasks, that can be distinguished by a process classifier. Such classifier can be a specific customer, a certain address or a particular operation. The properties of an activity sequence include the order, start time, duration and identification of activities.

This research has three objectives:

1. Develop a metric that allows to assess structural similarity among generic sequences of activities
2. Test if truck routing schedules can be described and compared based on structural similarity
3. Explore the conceptual applicability of structural similarity comparison in a business context

The results of the thesis can be summarized into the following four products:

1. **Proposed extension on (Damerau-)Levenshtein logic**
   In this thesis’ proposed metric, the (Damerau-)Levenshtein string edit logics have been further leveraged. Whereas the original metrics only deliver a distance value between two strings of characters, this thesis proposes a method to extract explicit edit operations to obtain one string of characters from the other. These operations are found by first calculating the original distance matrix, then to find a cheapest path using Dijkstra’s shortest path algorithm and last, to back-trace this path and extract which sections of the examined sequences correspond to which identified efficient edit operation.

2. **Proposed activity sequence similarity metric**
   The proposed activity sequence similarity metric uses a two-step approach to achieve a similarity score. The first step uses (Damerau-)Levenshtein text similarity logic with the extension described in Product 1 to extract the explicit edit operations to obtain one activity sequence from another. In this step, the edit operations are determined based on the position of the activities in the sequence, distinguished by their process classifier. The edit operations include matching, deletion, insertion, substitution and transposition of activities. Text similarity logic is used because text is composed of sequential (groups of) characters. Characters and activity process classifiers are both of type string; they inherently present
3. Descriptive similarity assessment of truck routing schedules
The proposed metric has been assessed in the comparison of truck routing schedules carried out by a large Dutch truck company. Seven logistics experts have done fifteen exercises where each exercise consisted of ranking five routing schedule on their similarity with a target schedule. This led to a total of 525 comparisons. The proposed metric has been trained with this data to replicate the expert opinion. The accuracy of the expert-trained proposed metric was slightly higher than if it was trained by random ranking as input in case of the Damerau-Levenshtein based logic. Overall, the accuracy of the proposed metric in the comparison of truck routing schedules may be considered low.

4. Relevance
The logic behind the proposed metric has the potential to improve activity sequences in a variety of business domains in the role of a similarity-based decision support tool (SDSS). The conceptual SDSS that is described looks for better activity sequences by exploring historical solutions with a better quality evaluation in the order of high to low similarity. This can potentially lead to faster problem solving and adequate improvement suggestions. Further, a machine learning extension is proposed that is able to generate new solutions by implementing feasibility rules extraction and genetic programming. The incorporation of genetic programming as a machine learning mechanism in the conceptual SDSS may reduce human bias, increase the extent of extracted features and automate sequence invention with human-competitive intelligence. The conceptual overview of such system is displayed below.
1. Initialize

Retrieve historical schedules

2. Learn

Extract feasibility rules
Simulate schedule

Start & select good feasible scenarios

Perform GA techniques (mutation, crossover)

A. Solve Problems

Business problem
Find high performing solutions to similar problems
Resume business

Repository of feasible schedules

Find similarities, better solutions

B. Improve new Schedules

New schedule
Simulate schedule
Release schedule
# Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Approach</td>
<td>2</td>
</tr>
<tr>
<td>1.4</td>
<td>Scope</td>
<td>3</td>
</tr>
<tr>
<td>1.5</td>
<td>Outline</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>Theoretical Background</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Similarity Metrics</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Routing Schedule Characteristics</td>
<td>6</td>
</tr>
<tr>
<td>2.3</td>
<td>Conceptual Framework</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>Methods</td>
<td>12</td>
</tr>
<tr>
<td>3.1</td>
<td>Identification Similarity Edit Operations</td>
<td>13</td>
</tr>
<tr>
<td>3.2</td>
<td>Temporal Similarity</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>The Proposed Metric</td>
<td>29</td>
</tr>
<tr>
<td>4</td>
<td>Results</td>
<td>31</td>
</tr>
<tr>
<td>4.1</td>
<td>Training Procedure</td>
<td>31</td>
</tr>
<tr>
<td>4.2</td>
<td>Data Selection and Scenario Description</td>
<td>32</td>
</tr>
<tr>
<td>4.3</td>
<td>Experiment Design</td>
<td>33</td>
</tr>
<tr>
<td>4.4</td>
<td>Assessment Results</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>Relevance</td>
<td>38</td>
</tr>
<tr>
<td>5.1</td>
<td>Conceptual Similarity-Based Decision Support Tool (SDSS)</td>
<td>38</td>
</tr>
<tr>
<td>5.2</td>
<td>Prototypical Example</td>
<td>39</td>
</tr>
<tr>
<td>5.3</td>
<td>Machine Learning SDSS Extension</td>
<td>42</td>
</tr>
<tr>
<td>6</td>
<td>Conclusion</td>
<td>44</td>
</tr>
<tr>
<td>6.1</td>
<td>The Proposed Metric</td>
<td>44</td>
</tr>
<tr>
<td>6.2</td>
<td>Scientific Added Value</td>
<td>44</td>
</tr>
<tr>
<td>6.3</td>
<td>Suitability of Proposed Metric in Truck Route Comparison</td>
<td>45</td>
</tr>
<tr>
<td>6.4</td>
<td>Relevance</td>
<td>45</td>
</tr>
<tr>
<td>6.5</td>
<td>Limitations and Suggestions for Further Research</td>
<td>46</td>
</tr>
<tr>
<td>A</td>
<td>Instruction Text Questionnaire</td>
<td>53</td>
</tr>
</tbody>
</table>
List of Figures

1.1 Research Approach ................................................. 3
1.2 UML Diagram of a Generic Activity Sequence ..................... 3

2.1 UML Diagram of a Routing Schedule .............................. 8
2.2 Conceptual Framework of a Similarity Assessment Between Activity Sequences $x$ and $y$ ................................................ 10
2.3 Example Identification Similarity Edit Operations .................. 10
2.4 Example Temporal Similarity Factors .............................. 11

3.1 Running Example Fictitious Routes ................................. 12
3.2 Identification Similarity Edit Operations Assessment Procedure .. 14
3.3 Levenshtein Procedure of Cost-Matrix Construction ................. 16
3.4 Dijkstra’s Shortest Path Through Levenshtein Distance Matrix .. 18
3.5 Dijkstra’s Shortest Path Through Damerau-Levenshtein Distance Matrix .. 19
3.6 Distance Matrix Graph Representation ............................ 20
3.7 Levenshtein Back-Trace Procedure of Operations With Cost One .. 21
3.8 Levenshtein Back-Trace Procedure of Operations With Cost Zero .. 22
3.9 Levenshtein Traced-Back Shortest Path Running Example .......... 24
3.10 Damerau-Levenshtein Traced-Back Shortest Path Running Example .. 24
3.11 Levenshtein Based Operations Running Example Graphical ........ 25
3.12 Damerau-Levenshtein Based Operations Running Example Graphical .. 26

4.1 Example Questionnaire Exercise .................................. 34

5.1 UML Diagram of a Generic Activity Sequence With Quality Evaluation .. 39
5.2 Graphical Representation of Prototypical Route Database ............ 41
5.3 Graphical Representation of SDSS Proposed Improvement Suggestion .. 42
5.4 SDSS and Machine Learning Ecosystem .......................... 43
List of Tables

2.1 Assessment Criteria Overview of Text Similarity Algorithms .................. 7
2.2 Comparative Routing Schedule Composition Operations ....................... 9

3.1 Running Example Fictitious Routes .................................................. 13
3.2 Levenshtein Distance Matrix Running Example $D$ ................................ 16
3.3 Damerau-Levenshtein Distance Matrix Running Example $DD$ ................. 18
3.4 Levenshtein Shortest Path Through Running Example Distance Matrix $D$ .. 24
3.5 Damerau-Levenshtein Shortest Path Through Running Example Distance Matrix $DD$ ................................................................. 25
3.6 (Damerau-)Levenshtein Operations Summary Running Example $x$ to $y$ .... 25
3.7 Temporal Similarity Match Operation Examined ................................... 28
3.8 Temporal Similarity Transpose Operation Examined ............................... 28

4.1 Maximum Difference Determination Between any two Rankings of Five Rankees 32
4.2 Assessed Training Scenarios ............................................................... 33
4.3 Overview of Similarity Factors Across Assessed Routing Schedule Pairs (n=525) 35
4.4 Training and Test Results ................................................................. 36
4.5 Training Accuracy Across Different Match Ratio Subsets Based on Leven-

shtein Logic ............................................................................................ 37

5.1 Prototypical Route Database ............................................................... 40
5.2 Prototypical Routes Similarity Scores .................................................. 41
Chapter 1

Introduction

In this thesis, a theoretical research on the comparison of generic activity sequences is presented. The motivation behind the research topic is described in section 1.1. In section 1.2, the research objectives resulting from the motivation will be treated. In section 1.3, the research approach towards these objectives is discussed. Further, in section 1.4, the scope of the presented research is provided. The chapter will end with the research outline in section 1.5.

1.1 Motivation

If one would be able to successfully describe structural differences between two activity sequences, not only in terms of a similarity value, but also in terms of explicit edit operations and temporal differences, several opportunities would arise. First of all, the mechanism would enable the use of a search engine within activity (sub-)sequences. Such a search engine could be beneficial in quick problem solving or achieving starting points for more complex activity sequence construction. The problem solving benefit can be achieved by searching for solutions previously made by experts to solve similar problems. For example, rescheduling is an iterative process that is performed when new information becomes available. According to Harjunkoski et al. (2014), deterministic rescheduling is a common task of human schedulers. If this behavior can be mimicked or supported, the effects and implementation speed of the rescheduling may benefit from the potential search engine features. The combinatorial problems found in planning and scheduling often become too complex to calculate an optimal solution, therefore expert input is often relied on (Garey et al., 1976; Blum and Roli, 2003). A practical example can be found in rescheduling trucks; when the time slot of a delivery is narrowed down or shifted towards a high traffic density time, the service level may be in danger. A solution that an expert could provide is to choose a different route that takes longer but has a more reliable delivery time slot. This solution could then be found by the proposed search engine to be similar to the current problem. The user could use this previously suggested expert solution to solve the current problem. This procedure would potentially save time, effort and expert personnel. Second of all, the mechanism would allow for efficient search in improvement suggestions. This efficiency may be reached by searching for better performance in activity sequences starting with sequences with high similarity to the sequence that is tested for availability of improvement suggestions. The potential of this mechanism is conceivably powerful as sequences of activities can be found in many business
domains. A truck routing schedule may be seen as a sequence of activities, where the delivery to a customer can be regarded as an activity. Therefore, the truck route domain can be a starting point to assess this potential. Few attempts have achieved success in developing activity sequence comparison metrics. Harjunkoski et al. (2014) indicate that it is important for business stakeholders to compare scheduling solutions and that the possibility to compare a schedule with its historic counterpart is indicated as extremely advantageous because it allows to evaluate the expected benefits of its implementation. Wall (1996) has proposed two similarity measurement methods to assess the distance between tasks performed in a certain sequence order. The first approach measures a Euclidean distance between any pair of tasks in the same position in the sequence. The second proposed distance measure is based on the mechanism in the edge recombination operator defined by Whitley et al. (1989). This mechanism holds the assessment of two incidence matrices that describe the precedence relationship between two schedules. Both approaches depend on identical activities in the two examined sequences and limit their comparison only on the identification of activities. Further, theoretical research is missing a formal procedure to describe structural differences between activity sequences in terms of activity identification and temporal differences.

1.2 Objectives

This master’s thesis has a theoretical nature. Its primary objective is to develop a metric that allows to assess structural similarity among generic sequences of activities from a descriptive point of view. The secondary objective is to test if truck routing schedules can be described and compared based on structural similarity and a descriptive point of view. The tertiary objective is to explore the conceptual applicability of structural similarity comparison in a business context.

1.3 Approach

The metric is constructed by synthesizing scientific approaches from text similarity metrics and temporal measurements. First, a literature review is performed on similarity metrics and their relevance in truck routing schedule comparison. Text similarity metrics are adopted because text has a comparable property of sequentiality as a sequence of process classifiers does. Second, text similarity metric logic is further extended to be able to structurally compare activity identifiers in two sequences. The resulting similarity logic will then be enhanced with a temporal activity assessment leading to this research’ proposed metric. Moreover, the proposed metric will be assessed in the area of truck routing schedule comparison with expert input. Further, the application possibilities in a business context are explored. An overview of the total research approach including the interrelatedness among the chapters is displayed in Figure 1.1.
1.4 Scope

The similarity measurement metric that is proposed in this research is a generic metric to compare two sequences of activities. It may be applied in any context where a sequence of activities is an essential unit of analysis. The common description of a generic activity sequence is displayed conform Unified Modeling Language (UML) in Figure 1.2. In this thesis, the applicability of the proposed metric is assessed in the context of truck routing schedules with data from a large Dutch truck operator.
1.5 Outline

The remainder of this thesis will be structured as indicated in the Research Approach in section 1.3. In chapter two, the theoretical background of similarity matching and routing schedule composition is discussed. In chapter three, the proposed metric is constructed. In the fourth chapter, the suitability of the proposed metric in the area of truck routing schedule comparison is treated. In the fifth chapter, the relevance of the proposed metric is explored in a business context. Finally, the thesis will close with a discussion on the set objectives with some suggestions for future research.
Chapter 2

Theoretical Background

In literature, few attempts have been identified in the area of similarity assessment among activity sequences. The aim of this research is to develop a metric that allows the similarity comparison between any two activity sequences from a descriptive perspective; without prior knowledge about their construction logic. In this thesis, the applicability of the metric in the comparison of truck routing schedules will be covered. Therefore, related work in both similarity measurement techniques and routing schedule characterization is treated. This leads to a conceptual framework that functions as the basis of the proposed metric. In the first section, similarity metrics are covered. In the second section, truck routing characteristics are further investigated. The chapter will close with the framework around which the proposed metric will be constructed.

2.1 Similarity Metrics

In this section, similarity metrics are discussed. The aim of the research is to describe activity sequences from a descriptive point of view. Therefore, the characteristics of an activity sequence that are visible on the surface are examined. First, a distinction is made between characteristics that are process classifiers and characteristics that can be compared numerically. Numerically comparable characteristics are measurable, such as the start time of an activity. These characteristics are further referred to as temporal elements of an activity sequence. Process classifiers are defined as classifiers of an activity and describe the identification of the specific activity. These classifiers can be for instance the visit of a specific customer, a stay at a certain address or the execution of a particular operation. These classifiers are further referred to as identification elements of an activity in a sequence. The descriptive similarity assessment of the identification elements is limited to comparing the unique process identifiers of the activities.

Unique identifiers can be composed of any combination of characters: letters, numbers or symbols. The main reason for the issues raised with similarity computation among (categorical) identifiers is that they are not inherently ordered (Boriah et al., 2008). In this research, character-based similarity measures are investigated as they allow the comparison of activities in terms of their unique identifiers. This is particularly valuable because the activity sequences are compared from a descriptive perspective; the underlying decision variables are unknown. Moreover, character similarity metrics compare (sub-)strings of sequential characters. In similarity determination of activity sequences, process identifiers can
be described as (sub-)strings of sequential characters. This parallel ratifies the incorporation of character-based similarity metrics in the construction of a generic similarity metric for activity sequences.

In the remainder of this section, the character-based similarity metrics as provided by Gomaa and Fahmy (2013) are investigated on their similarity determination mechanism. In the conceptual framework at the end of the chapter, the measurement method that aligns most with the objectives of this study will be incorporated in the conceptual framework. In Table 2.1, an overview is provided of the treated text similarity measurement algorithms and the mechanisms that are are used in the assessment.

The LCS (Longest Common Subsequence) metric measures the distance between two strings based on the length of contiguous chain of characters that occur in both strings. It uses matching, deletion and insertion operations to find the similarity score. The Levenshtein distance gives the minimum number of insertions, deletions and substitutions required to transform one string into the other (Levenshtein, 1966). The Damerau-Levenshtein distance also allows for transposing two adjacent characters. Weights can be added to the operations insertions, deletions and substitutions. The Jaro distance (Jaro, 1995) allows for a more flexible number of transpositions to measure the similarity. Two matched characters are referred to as “matching” if their respective position in string x and string y is not farther than \( \left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1 \). Therefore, the operations that are allowed in the similarity measure are matching and transposing. The Jaro-Winkler distance (Winkler, 1990) is an extension of the Jaro distance, giving a higher similarity score to strings that match from the beginning. The Needleman-Wunsch is relevant for strings of equivalent lengths. It has a background in bioinformatics. The allowed operations to measure the similarity are matches, insertions, deletions and mismatches. Because mismatches are allowed, rather than substitutions, the lengths of the examined strings should be of equal length. The Smith-Waterman measure (Waterman et al., 1976) has a background in biological sequence matching. It is a form of dynamic programming that may be useful in application areas where patterns are suspected to contain regions of similarity. The approach is similar to Needleman-Wunsch and the result is some global alignment score. The N-gram measure (Barrón-Cedeño et al., 2010) is often found in plagiarism detection. The measure is based on the search of an n-gram (number of characters, syllables or words) in a provided set of texts. Further, this method is applied by Google allowing to search strings in all available book sources printed from 1500 and onwards (Books.Google.com, 2010). The measure looks for sequences of co-occurring words in a text. The score is calculated by dividing the number of matched n-grams by the total number of n-grams. Non-matching sequences can therefore be disregarded. This operation can be referred to as deleting or inserting items.

2.2 Routing Schedule Characteristics

In this research, a similarity measurement metric for sequences of activities is proposed. The proposed metric has a descriptive nature; the rationale underlying the construction of such sequence of activities is assumed as given. The applicability of the proposed metric is assessed in the context of comparing truck routing schedules. Therefore, the theoretical descriptive characteristics and some routing schedule comparison heuristics are further investigated. The aim of this section is to study operations in routing composition that are similar to operations in text comparison assessment.
### Table 2.1: Assessment Criteria Overview of Text Similarity Algorithms Suggested by Gomaa and Fahmy (2013)

<table>
<thead>
<tr>
<th>Criterium / Algorithm</th>
<th>LCS Damerau-Levenshtein</th>
<th>Jaro-Jaro-Winkler</th>
<th>Needleman-Wunsch</th>
<th>Smith-Waterman</th>
<th>N-gram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matching</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Transposing</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deleting</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Inserting</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Requiring equal lengths</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 2.2.1 Theoretical descriptive characteristics of routing schedules.

In literature, much attention has been paid to the assessment or optimization of a specific schedule. An approach to analyze the descriptive properties of a routing schedule is to look at the output or representation of such optimization, from a descriptive point of view. The aim of this section is to identify whether the characteristics of routing schedules may be similar to a generic activity sequence. According to Desrochers et al. (1990), solutions in the vehicle routing scene are mostly related to the objectives travel and service time. Hägerstrand (1970) has developed a unified paradigm to analyze complex travel behavior. A specific pattern of activities is regarded as the solution of an allocation problem, involving limited resources of time and space. Such solution may be similar as a solution of a professional planner that is allocating activities according to constraints. Solomon (1987) concluded that in a vehicle routing environment, the waiting and loading time as a result of arriving too soon or too late at a customer are important performance measures of any routing schedule. These properties of a visit can be referred to as a shift in starting time and service time. Further, Desrochers et al. (1990) have identified that the standard objectives found in the vehicle routing and scheduling literature are the minimization of the total travel and service time and minimization of the span of a solution. Harris and Ioannou (1998) have identified starting time, ending time and duration of an activity necessary performance characteristics for a multi-unit project schedule. Chang et al. (2003) have proposed a mathematical model for real-time vehicle routing problems. The solution of the model incorporates a number of stops performed by a certain type of vehicle. Therefore, it can be concluded that next to the occurrence of a stop at a certain customer, both the start time and service time, or duration length, of an activity in a routing schedule are important characteristics of a truck routing schedule. More specifically, a routing schedule may be defined as a number of stops at different customers along a time line performed by a type of vehicle where both the service and travel time are incorporated. In Figure 2.1, the general description of such routing schedule analogue to the example of an activity sequence in Figure 1.2 is displayed. The address of a specific customer functions as the unique identifier of a truck stop. This representation leads to construction of Hypothesis 1:

**H1:** Truck routing schedules can be described as generic sequences of activities.
2.2.2 Theoretical comparison heuristics of truck routing schedules.

In the next part, theoretical heuristics in truck routing schedules that show traits of comparison among schedules are treated. The objectives and mechanisms of these heuristics are relevant for the assessment of the applicability of the comparison of the proposed activities sequence comparison metric.

Solomon (1987) identified different heuristics that are used in routing planning and have some traits of comparative improvements. One of the heuristics classes they identified is the class *insertion heuristics*, where at every iteration, the effect of inserting a customer is evaluated. In the class, they also find heuristics that shift the start of the servicing time of the inserted customer. The second approach that is addressed is selecting specific customers for the insertion that minimize a measure of total route distance and time. The third heuristic that is assessed is the class insertion heuristics, which is related to inserting customers from a time-oriented, nearest-neighbor point-of-view. Solomon (1987) expected that the insertion heuristics class reduces the waiting time in the schedules produced by these heuristics to be significantly lower than those produced by distance-driven criteria.

Cordeau et al. (2002) express a classical vehicle routing problem (VRP) as a graph of customers with weights on the edges in between. The Taburoute method as proposed by Gendreau et al. (1994) has been identified as the best meta-heuristic for the VRP. The method focuses on the exchange of a customer from one route to another, or the deletion of a customer in one route and inserting it in another. Lin (1965) attempted to improve a vehicle routing problem by removing some customers and reconnecting the remaining chains along the routing. This can be referred to as re-evaluating performance after deletion.

A different area of interest can be described as multi-route edge exchanges for the VRP (Laporte et al., 2000). Van Breedam (1994) for example, discusses the potential of string cross, string exchange, string relocation and string mix in local improvements. These operations can be described as transposing and substituting customers to achieve better performance. Further, some implementations of the well-known *sweep* algorithm (Shamos and Hoey, 1976) accommodate the exchange of vertices between adjacent clusters. These prop-
erties have been investigated by Gillett and Miller (1974) and Wren and Holliday (1972) and many others, and can be referred to as transposition operations. Inserting, deleting, substituting customers from one route to the next have been shown to be relevant in improving one routing schedule to another. Therefore, these operations are regarded as important comparison elements between two routing schedules. Besides, the transposition or relocation of elements within a routing schedule has been shown to have potential for local improvements. In Table 2.2, an overview is provided of the investigated operations in comparative routing schedule composition throughout the treated articles.

Table 2.2: Comparative Routing Schedule Composition Operations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deleting</td>
<td>Gendreau et al. (1994); Cordeau et al. (2002); Lin (1965)</td>
</tr>
<tr>
<td>Inserting</td>
<td>Solomon (1987); Cordeau et al. (2002)</td>
</tr>
<tr>
<td>Substituting</td>
<td>Van Breedam (1994); Laporte et al. (2000)</td>
</tr>
<tr>
<td>Transposing</td>
<td>Van Breedam (1994); Wren and Holliday (1972); Gillett and Miller (1974)</td>
</tr>
</tbody>
</table>

2.3 Conceptual Framework

In the previous sections, different similarity metrics have been investigated. Also, the possible resemblance of a routing schedule and a generic activity sequence has been treated. Moreover, text similarity operations seem to be comparable with routing schedule improvement operations. Therefore, the comparison of routing schedules may be appropriate when applying a suitable text similarity metric. Further, in this research, a new metric is proposed and assessed in the context of routing schedules.

According to the results of Xing et al. (2003), the expected accuracy of a distance metric may be low compared to assessments with much available side information. Truck routing schedule composition is subject to much external information such as expected weather, traffic and seasonality influences. For instance, heavy rainfall decreases freeway capacity by 25-30% (Smith et al., 2004). Since the proposed metric has a purely descriptive nature, disregarding all side information, the expected accuracy may be low. However, reason has been shown to assume that a routing schedule can be described as a generic activity sequence. Therefore, the value of the proposed metric may have a better than random value, purely based on descriptive assessment. This conjecture is tested with Hypothesis 2:

**H2:** Comparing routing schedules based on similarity metrics results in a better than random comparison accuracy.

A similarity score between activity sequences \( x \) and \( y \) may be found by synthesizing both the similarity in terms of the identification of activities and the temporal differences between corresponding activities. The similarity in terms of the identification of activities is referred to as the identification similarity. The operations to obtain sequence \( y \) from sequence \( x \), solely looking at the identification of activities are referred to as identification similarity edit operations. The temporal assessment consists of a further investigation of a matched or transposed activity in the sequence. The conceptual framework of the similarity assessment between generic activity sequences \( x \) and \( y \) is displayed in Figure 2.2.
In Figure 2.3, an example of a series of identification similarity edit operations of two fictitious routes $x, y$ is displayed. The identification similarity edit operations are based on the Levenshtein algorithm and indicate the lowest total identification similarity edit cost to obtain route $y$ from route $x$. In the figure, it is shown that Philadelphia and Baltimore have been found to be of identification similarity edit operation match. Furthermore, in Figure 2.4, the temporal similarity factors start time difference (A) and duration difference (B) among the matched locations Baltimore and Philadelphia are presented.
Figure 2.4: Example Temporal Similarity Factors (A=Duration Difference, B=Start Time Difference)
Chapter 3

Methods

In the forthcoming section, the proposed similarity measurement method is discussed. The proposed metric is achieved by synthesizing identification similarity edit operations and temporal similarity factors of activities that are identified as identical. The synthesis is done by integrating two steps of measurements. In the first step, identification similarity edit operations between activity sequences are investigated with logic lend from the (Damerau-)Levenshtein text similarity metrics. This approach results in the extraction of edit operations that show the lowest edit costs between the two examined sequences. In the second step, these edit operations are investigated in a temporal way, based on starting time and duration. The occurrence ratios of the edit operations and the temporal factors are referred to as similarity factors.

In the first section, the assessment of the identification similarity edit operations is explained. In the second section, the temporal similarity factors among the identically identified activities is discussed. The chapter will close with the proposed metric. The approach will be tested in the next chapter with routing schedule data from a truck transportation company. Therefore, and to clarify the different steps in the metric, the explanation of the metric is guided by a running example. The presented running example is the comparison of two fictitious routes $x, y$. In Figure 3.1, a graphical representation of the routes is displayed. The data about the fictitious routes is displayed in Table 3.1.

![Figure 3.1: Running Example Fictitious Routes](image-url)
Table 3.1: Running Example Fictitious Routes

<table>
<thead>
<tr>
<th>Stop no.</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Route x</td>
<td>New York</td>
<td>Philadelphia</td>
<td>Washington</td>
<td>Baltimore</td>
<td></td>
</tr>
<tr>
<td>Start time</td>
<td>08:00</td>
<td>10:10</td>
<td>13:15</td>
<td>15:10</td>
<td></td>
</tr>
<tr>
<td>End time</td>
<td>08:20</td>
<td>10:15</td>
<td>14:15</td>
<td>15:30</td>
<td></td>
</tr>
<tr>
<td>Travel time</td>
<td>1:50</td>
<td>3:00</td>
<td>00:55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Route y</td>
<td>Philadelphia</td>
<td>Bel Air</td>
<td>Baltimore</td>
<td>Washington</td>
<td>Salisbury D.C.</td>
</tr>
<tr>
<td>Start time</td>
<td>08:00</td>
<td>10:35</td>
<td>10:40</td>
<td>12:00</td>
<td>15:45</td>
</tr>
<tr>
<td>End time</td>
<td>08:30</td>
<td>10:30</td>
<td>10:55</td>
<td>13:00</td>
<td>16:10</td>
</tr>
<tr>
<td>Travel time</td>
<td>2:05</td>
<td>00:10</td>
<td>1:05</td>
<td>2:45</td>
<td></td>
</tr>
</tbody>
</table>

3.1 Identification Similarity Edit Operations

In chapter 2, the character-based similarity algorithms that were investigated by Gomaa and Fahmy (2013) have been assessed in their applicability in routing schedule characteristics comparison. Because of its richness in comparison with other similarity metrics, the Damerau-Levenshtein logic is further built upon in this research. However, the Damerau-Levenshtein logic is not a similarity metric, in the sense that it violates the condition of triangle-inequality. According to Tversky and Gati (1982), this condition may be the most basic property of a metric model. It requires that the direct path from a to c will not exceed the indirect path through b. Further, this criterion allows to quickly reach conversion in a search with the concept of closed sets. It is achieved since triangle inequality allows to define the topology in a metric space. The purpose of the presented research is to develop a scientifically valid metric. Apart from reaching this purpose, the algorithm inherently poses some challenges in the memoryless manner it was originally constructed. Therefore, both an adapted version of the Damerau-Levenshtein algorithm and its predecessor, the Levenshtein algorithm are evaluated in this paper. The Damerau-Levenshtein algorithm is investigated to compare the performance in similarity matching with the Levenshtein algorithm. One might expect a better Damerau-Levenshtein performance because of the allowed transposition properties of the algorithm, which has been indicated by Cordeau et al. (2002); Van Breedam (1994) to play a role in the composition of routing schedules.

The objective of this section with respect to proposed logic is to extract a set of efficient identification similarity edit operations to transform activities or customers sequence x into sequence y. The operations allowed when using Levenshtein logic are matching, deleting, inserting and substituting. The Damerau-Levenshtein uses a second round of assessment to identify transposition operations in similarity matching. First, the Levenshtein logic is explained. Thereafter, the Damerau extension will be treated.

In the ordinary form of Levenshtein similarity matching, the logic is used to extract a single number that represents the distance between two strings. The algorithm runs through the two strings by the use of matrix calculations where no decisions or path directions are stored. Therefore it can be said that the Levenshtein metric is not suitable to extract useful information due to its memoryless properties. In this study, a novel approach is proposed to extract efficient edit operations to obtain string y from x. The approach leverages the Levenshtein logic by first creating a cost matrix, then to find the cheapest path through
this matrix and last to identify the edit operations associated with this route. The cheapest
path is achieved using the Dijkstra’s algorithm (Dijkstra, 1959). The Dijkstra algorithm is
used for finding shortest paths from a single node to any other one. This is also referred to
as Single Source Shortest Path (SSSP) (Skiena, 1990). The algorithm finds the lowest cost
path through a cost matrix where incremental costs are regarded as distance. In essence,
the found Dijkstra’s shortest path through the distance matrix is not the shortest path,
but a path with the lowest possible cumulative edit cost. This extension on Levenshtein
algorithm allows to make similarity factors explicit rather than only calculating the final
similarity score. The Damerau-Levenshtein algorithm offers more memoryless properties, as
after the transposition of two elements in one string to two elements in another investigated
string, only the latter of the two is taken into account when further calculating towards the
edit distance.

Conclusively, both algorithms are constructed in such way that calculations earlier in
the string are no longer of value, which suffices the objective of calculating a final edit
distance. In the presented research, the possibilities of overcoming the properties of the
aforementioned memorylessness are further explored. Additionally, novel iterations on both
algorithms are presented to leverage the opportunities in the mechanism of both similarity
matching algorithms.

A schematic representation of the presented identification similarity edit operations as-
assessment that allows to extract these explicit operations to obtain sequence $y$ from $x$ using
both Levenshtein and Damerau-Levenshtein algorithm together with Dijkstra’s shortest path
algorithm is displayed in Figure 3.2. In the following sections, the Levenshtein logic, the
Damerau extension, the shortest path determination and a proposed back-tracing procedure
to extract efficient edit operation will be explained.

![Figure 3.2: Identification Similarity Edit Operations Assessment Procedure](image)

### 3.1.1 Create Levenshtein distance matrix between sequences $x, y$. 

Consider $x, y$ being two schedules with lengths of successive activities $|x|$ and $|y|$. We use $x_i$ to
denote the $i^{th}$ element of $x$. Each activity in $x$ has a start time and a duration. We use $ST_{i,x}$ to
denote the start time of the $i^{th}$ element in $x$. We use $DT_{i,x}$ to denote the duration of the
$i^{th}$ element in $x$. We use $y_j$ to denote the $j^{th}$ element of $y$. Each activity in $y$ has
a start time and a duration. We use $ST_{j,y}$ to denote the start time of the $j^{th}$ element in $y$.
We use $DT_{j,y}$ to denote the duration of the $j^{th}$ element in $y$.

The term activity is chosen because any planning consists of elements that are charac-
terized by a start and end time. The identification similarity edit operations between these
schedules can be measured using an appropriate string edit metric. The default Levenshtein
cost metric is based on matrix calculations and is usually only applied to calculate a final
similarity score. The scores are achieved by calculating distances to adjacent cells from column to column. It can be said that the algorithm is memoryless in the sense that by default, the lowest cost edit operations are not stored or documented. The logic to construct the Levenshtein distance cost matrix $D$ is displayed in Algorithm 1 (Levenshtein, 1966).

**Algorithm 1** Levenshtein Edit Distance

1: **procedure** Levenshtein edit distance
2: let $D[0 : |x|, 0|y|]$ be a 2d matrix with $|x| + 1$ rows and $|y| + 1$ columns
3: for $i$ in 1 to $|x|$ do
4:   $D[0 : |x|, 0] ← i$
5: for $j$ in 1 to $|y|$ do
6:   $D[0, 0 : |y|] ← j$
7: for $i$ in 1 to $|x|$ do
8:   for $j$ in 1 to $|y|$ do
9:     if $x_i = y_j$ then
10:        cost ← 0
11:     else
12:        cost ← 1
13:     $D_{i,j} ← \min(D_{i−1,j} + 1, D_{i,j−1} + 1, D_{i−1,j−1} + cost)$

The logic implies that for the Levenshtein logic, the construction of the distance cost matrix is constructed by minimizing costs from adjacent cells, as displayed in Figure 3.3. In words, the operations that are performed in the Levenshtein algorithm include:

1. A match ($m$) is a pair of indices $(i,j)$, indicating that $y_j$ is obtained from $x_i$ by matching, associated with cost $D_{i,j} = D_{i−1,j−1}$. We define $M$ as the set of all matches produced by the algorithm.

2. A deletion ($d$) is a pair of indices $(i,j)$, indicating that $y_j$ is obtained from $x_i$ by deleting, associated with cost $D_{i,j} = D_{i−1,j} + 1$. We define $D$ as the set of all deletions produced by the algorithm.

3. An insertion ($i$) is a pair of indices $(i,j)$, indicating that $y_j$ is obtained from $x_i$ by inserting, associated with cost $D_{i,j} = D_{i,j−1} + 1$. We define $I$ as the set of all insertions produced by the algorithm.

4. A substitution ($s$) is a pair of indices $(i,j)$, indicating that $y_j$ is obtained from $x_i$ by substituting, associated with cost $D_{i,j} = D_{i−1,j−1} + 1$. We define $S$ as the set of all substitutions produced by the algorithm.
Figure 3.3: Levenshtein Procedure of Cost-Matrix Construction

For the running example, the Levenshtein algorithm result in distance matrix $D$ in Table 3.2. The algorithm starts by filling row and column zero with integers starting with value 0 in cell $(0,0)$. Then, the distance matrix is filled by assessing costs in individual cells in the order of the columns. As is described in Algorithm 1 and the Figure 3.3, the cell gets assigned the minimum value among the cells with direction up, up left and left plus a cost factor. A minimum found in the adjacent cells with directions up (deletion) and left (insertion) have cost one. The diagonal step has cost one in case of a substitution and cost zero in case of a match. For instance, in cell $D(1,1)$, Philadelphia and New York are not a match, therefore the lowest edit cost in $D(1,1)$ is equal to $\min(D(0,1) + 1, D(1,0) + 1, D(0,0) + 1 = 1)$.

Table 3.2: Levenshtein Distance Matrix Running Example $D$

<table>
<thead>
<tr>
<th></th>
<th>New York</th>
<th>Philadelphia</th>
<th>Bel Air</th>
<th>Baltimore</th>
<th>Washington</th>
<th>Salisbury</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3.1.2 Extend Levenshtein distance matrix with Damerau logic.

By adding the Damerau extension, the property of transposition options between two adjacent items in the sequence is added in the assessment of similarity (Damerau, 1964). For clarity purposes, the distance matrix where the Damerau extension is incurred in the construction of the distance matrix is referred to as $DD$. In Algorithm 2, the Damerau Extension is displayed.

**Algorithm 2** Damerau Extension on Levenshtein Edit Distance

1: procedure DAMERAU TRANSPOSE ACTIONS
2: $\text{if } i > 1 \text{ and } j > 1 \text{ and } x_i = y_{j-1} \text{ and } x_{i-1} = y_j \text{ then}$
3: $DD_{i,j} \leftarrow \min(DD_{i,j}, DD_{i-2,j-2}) + 1$

The Damerau extension results in a set of transposable cells $DD_{i,j}$ and $DD_{i-1,j-1}$ where
$DD_{i,j}$ consists of the minimal value of the current investigated cell or the combination $DD_{i-2,j-2}$. In its original purpose, where only the final edit distance number is evaluated, this procedure suffices the purpose. More specifically, $D_{i-1,j-1}$ is not changed because it is not necessary if $\min(D_{i,j}, D_{i-2,j-2}) + 1$ fills in $D_{i,j}$ with which the calculation until the cell in the bottom right corner is reached.

In this research, it is proposed to back-trace through the cost matrix to extract explicit edit operations. Therefore, the value in $DD_{i-1,j-1}$ is important because it is re-evaluated during the back-tracing. The Damerau charges cost 1 to the transpose action. This is the reason that Algorithm 2 is extended by setting the $DD_{i-1,j-1}$ that represents the first position of the transposed adjacent items to $DD_{i-2,j-2}$. Therefore, the total cost of the transposition is set to 1. This leads to the construction of Algorithm 3 when preparing the cost matrix with Damerau extension for Dijkstra’s algorithm. The proposed cost change for $DD_{i-1,j-1}$ has been shaded gray. Further, the cost of operations can be changed according to their importance in the context of the analysis. In this research, a deviation from the proposed nuance in the Damerau extension is accepted where the cost of a transpose is set to zero, to improve the attractiveness of following the transpose route. Besides, the costs in deciding on the route has an influence of deciding which operation is used to obtain sequence $y$ from $x$ and has no direct influence on the similarity value, as it is parametrized by expert input or historical data.

**Algorithm 3** Damerau-Levenshtein Edit Distance

1:  procedure Damerau-Levenshtein edit distance  
2:  let $DD[0 : |x|, 0 : |y|]$ be a 2d matrix with $|x| + 1$ rows and $|y| + 1$ columns  
3:  for $i$ in 0 to $|x|$ do  
4:      $DD[0 : |x|, 0] \leftarrow i$  
5:  for $j$ in 0 to $|y|$ do  
6:      $DD[0, 0 : |y|] \leftarrow j$  
7:  for $i$ in 1 to $|x|$ do  
8:      for $j$ in 1 to $|y|$ do  
9:          if $x_i = y_j$ then  
10:              cost $\leftarrow 0$  
11:          else  
12:              cost $\leftarrow 1$  
13:              $DD_{i,j} \leftarrow \min(DD_{i-1,j} + 1, DD_{i,j-1} + 1, DD_{i-1,j-1} + \text{cost})$  
14:          if $i > 1$ and $j > 1$ and $x_i = y_j-1$ and $x_{i-1} = y_j$ then  
15:              $DD_{i,j} \leftarrow \min(DD_{i,j}, DD_{i-2,j-2} + \text{cost})$  
16:              $DD_{i-1,j-1} \leftarrow DD_{i-2,j-2}$

This implies that for the Damerau extension, the transposition operation can be described as:

5. A transposition ($t$): two pairs of elements with adjacent indices $(i,j-1)$, $(i-1,j)$ indicating that $y_j$ and $y_{j-1}$ are obtained from $x_{i-1}$ and $x_i$ by transposing, associated with cost $DD_{i,j} - DD_{i-2,j-2} + \text{cost}$.

For the running example, the algorithms result in distance matrix $DD$ in Table 3.3 for the Damerau-Levenshtein algorithm. The transposition costs are shaded gray.
Table 3.3: Damerau-Levenshtein Distance Matrix Running Example $DD$

<table>
<thead>
<tr>
<th></th>
<th>Phila-delphia</th>
<th>Bel Air</th>
<th>Baltimore</th>
<th>Washington</th>
<th>Salisbury</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Philadelphia</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Washington</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Baltimore</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

3.1.3 Find shortest path through cost matrix using Dijkstra’s algorithm.

Now that the cost matrices have been constructed, the operations that require the lowest edit cost to transform sequence $x$ into sequence $y$ can be further investigated. The lowest cost is defined as the minimum cost to go from the upper left corner in the cost matrix, to the bottom right. For this purpose, a shortest path algorithm is used. First, the cost matrix is transformed into a directed graph, where the edges are defined as the operations going right, slant right down and directly down in the $D$ or $DD$ cost matrix. For the lower row, only edges to the right are allowed and for the right column, only operations down are allowed. The edges represent the (cumulative) costs of the operations. The vertices represent customers that are visited, or activities that are performed. If transposition is allowed and such opportunity has been spotted, the edges right and directly down from the first vertex of the pair of transposable vertices are removed. This correction is made to force the shortest path algorithm to exploring the transpose opportunity when it has arrived at the low cost of following these edges.

For the running example, the directed graphs and shortest paths are displayed in Figures 3.4 and 3.5 for the Levenshtein and Damerau-Levenshtein logic respectively. The output of this procedure is a set of steps that is most efficient in obtaining sequence $y$ from sequence $x$ through the distance cost matrix.

Figure 3.4: Dijkstra’s Shortest Path Through Levenshtein Distance Matrix
The Dijkstra algorithm is used for finding shortest paths from a single node to any other one. This is also referred to as Single Source Shortest Path (SSSP) (Skiena, 1990). The Dijkstra algorithm is greedy, which implies that all possible paths through the graph are examined. The time complexity of computing all routes from a single vertex is equal to $n^2 + m$ with $n$ the number of vertices, and $m$ the number of edges (Magzhan and Jani, 2013). Although other algorithms might be faster, the Dijkstra algorithm is used since it examines all routes, therefore eliminating the chance to end in a sub-optimal path. When comparing sequences of activities or serviced customers, it might be the case that some low cost edges (match or transpose operations) are found at two opposite sides of the two sequences. In order to not oversee these patterns, all paths must be examined. One of the research’s objectives is to assess the application of the proposed logic in a business context. One such application may be in a decision support tool. It is important that Dijkstra’s algorithm finds the exact same path every time that the same operations matrix is examined. The Dijkstra algorithm finds the paths by calculating the edge weight to all adjacent vertices. The calculations are done in the ascending order of the vertex numbers, or a clockwise order starting at the direction down. The vertices are numbered according to their position inside the column before moving to the next column. For instance in Figure 3.6 vertex 12, the algorithm would first measure the weight on edge 12-13, then 12-17 and then 12-18. Therefore, in the case that more different paths would yield the same distance cost as the lowest, the same path would be found in every run.
3.1.4 Back-trace edit operations from cost matrix.

Now the $D$ and $DD$ cost matrices have been constructed, the most efficient edit operations can be obtained by back-tracing the Dijkstra’s shortest path through the distance matrices. For this purpose, the identified Dijkstra’s shortest path $P$ is traced-back from the bottom right corner to the upper left corner.

Let $P$ represent a sequence of tuples $T$. Each tuple $T$ represents a coordinate in matrix $D$ or $DD$. For example: $P = (1,1), (1,2), \ldots, (4,4)$. As convention, we refer to $P_k$ as the $k^{th}$ element of $P$ and $T_k$ as the $k^{th}$ element of $T$, such that, for example, $P_1^2$ is the first coordinate of the second tuple in $P$, i.e. $P_1^2 = 1$ in the example. Let $O$ represent a sequence of $k$ elements. Each element $O_k$ represents an identification similarity edit operation found by back-tracing Dijkstra’s shortest path through matrix $D$ or $DD$. For example: $O = \langle \text{match}, \text{insert}, \text{delete} \rangle$. As convention, we refer to $O_k$ as the $k^{th}$ element of $O$, such that, for example, $O_2$ is the second element in $O$, i.e. $O_2 = \text{insert}$ in the example.

Algorithm 4 describes the proposed back-tracing procedure following Dijkstra’s shortest path in distance cost matrix $D$. This will result in an output with efficient, explicit edit operations to obtain sequence $y$ from sequence $x$. The back-tracing procedure of the optional Damerau extension has also been defined. This extension allows the matching of a transposition between two adjacent items in sequences $x$ and $y$.

The proposed back-tracing procedure of the Levenshtein logic starts with reversing the search direction, or arrows in Figure 3.3. Then two distinctions are made in order to uncover the appropriate edit operation from the combination of the cost matrix and the found shortest path. The first distinction holds when the cost difference associated with one step on the reversed Dijkstra’s shortest path through the distance matrix is equal to one. The second distinction holds when the cost difference associated with one step on the reversed Dijkstra’s shortest path through the distance matrix is equal to zero.
First, when \( D_{p_1^k, p_2^k} - D_{p_1^{k-1}, p_2^{k-1}} = 1 \), an edit operation has been found on Dijkstra’s shortest path with edit cost one. This edit cost occurs in the case of a delete, an insert or a substitute operation. The determination of operation \( O_k \) distinguished with edit cost one can therefore be traced-back along the shortest path according to the logic as displayed in Figure 3.7.

Second, when \( D_{p_1^k, p_2^k} - D_{p_1^{k-1}, p_2^{k-1}} = 0 \), an edit operation has been found on Dijkstra’s shortest path with edit cost zero. An edit cost of zero occurs in the case of a match. The match is the first operation in a sequence, or preceded by either a match or a substitute (diagonal path movement), a delete (vertical path movement) or an insert (horizontal path movement). In the case of a delete or an insert operation preceding a match in \( O_k \), the cost of this operation has already been made in operation \( O_{k-1} \), or when the Dijkstra path moves from \( D_{p_1^{k-2}, p_2^{k-2}} \) to \( D_{p_1^{k-1}, p_2^{k-1}} \). The operation thereafter is a match, and consequently of edit cost zero. Therefore, in the case of a match preceded by an insert or a delete operation, the determination of \( O_k \) and \( O_{k-1} \) should be considered jointly. As is shown in Algorithm 4 lines 9:16, the traceback is then picked up again at element \( k - 2 \). If the match is preceded by a substitute or a match, the traceback continuous at element \( k - 1 \). Last, when the first operation in the sequence is a match, it is found in \( D_{p_1^1, p_2^1} \) associated with \( O_2 \). When \( k = 2 \) is a match, \( D_{p_1^2, p_2^2} \) would be zero according to the Levenshtein logic. Therefore, the procedure as described in Algorithm 4 lines 6:8 would assign operation match, reduce \( k \) and terminate the search since \( k \leq 1 \). Conclusively, the determination of operations \( O_k \) and \( O_{k-1} \), with \( O_k \) identified as a match, can therefore be traced-back along the shortest path according to the logic as displayed in Figure 3.8.
Figure 3.8: Levenshtein Back-Trace Procedure of Operations With Cost Zero
Algorithm 4 Levenshtein Cost Traceback Following Dijkstra’s Shortest Path

1: procedure BACK-TRACE SHORTEST PATH THROUGH DISTANCE MATRIX
2: Input: Distance cost matrix $D$ or $DD$, Dijkstra’s shortest path $P$
3: Output: Sequence of operations $O$, $|O| = |P|$
4: $k = |P|$
5: while $k > 1$ do
6:   //Match preceded by match or substitution or non-preceded
7:     if $D_{P_k^1, P_k^2} - D_{P_{k-1}^1, P_{k-1}^2} = 0$ and $P_{k-1}^1 = P_{k-1}^1$ and $P_{k-1}^2 = P_{k-2}^2$ then
8:       $O_k \leftarrow$ match
9:       $k = k - 1$
10: else if $D_{P_k^1, P_k^2} - D_{P_{k-1}^1, P_{k-1}^2} = 0$ and $P_{k-1}^1 = P_{k-1}^1$ and $P_{k-1}^2 = P_{k-2}^2$ then
11:       $O_k \leftarrow$ match
12:       $k = k - 1$
13: else if $D_{P_k^1, P_k^2} - D_{P_{k-1}^1, P_{k-1}^2} = 0$ and $P_{k-1}^1 = P_{k-1}^1$ and $P_{k-1}^2 = P_{k-2}^2$ then
14:       $O_k \leftarrow$ match
15:       $k = k - 2$
16: //Match preceded by deletion
17: else if $D_{P_k^1, P_k^2} - D_{P_{k-1}^1, P_{k-1}^2} = 1$ and $P_k^1 = P_{k-1}^1$ and $P_k^2 = P_{k-1}^2$ then
18:       $O_k \leftarrow$ delete
19:       $k = k - 2$
20: else if $D_{P_k^1, P_k^2} - D_{P_{k-1}^1, P_{k-1}^2} = 1$ and $P_k^1 = P_{k-1}^1$ and $P_k^2 = P_{k-1}^2$ then
21:       $O_k \leftarrow$ delete
22:       $k = k - 1$
23: //Deletion, insertion or substitution
24: else if $D_{P_k^1, P_k^2} - D_{P_{k-1}^1, P_{k-1}^2} = 1$ and $P_k^1 = P_{k-1}^1$ and $P_k^2 = P_{k-1}^2$ then
25:       $O_k \leftarrow$ substitute
26:       $k = k - 1$
27: Optional Damerau extension
28: else if $DD_{P_k^1, P_k^2} - DD_{P_{k-1}^1, P_{k-1}^2} = 0$ and $DD_{P_k^1, P_k^2} - DD_{P_{k-1}^1, P_{k-1}^2} = 0$ and
29:       $P_{k-1}^1 = P_{k-1}^1$ and $P_{k-1}^2 = P_{k-2}^2$ and $P_{k-2}^2 = P_{k-1}^2$ and $P_{k-2}^1 = P_{k-1}^1$ and
30:       $x_{P_{k-1}^1} = y_{P_{k-2}^2}$ then
31:       $O_k \leftarrow$ transpose
32:       $k = k - 2$
33: The result of executing the algorithm in the running example yields explicit edit operations to obtain sequence $y$ from sequence $x$. The explicit operations can be identified and displayed in the back-traced Dijkstra’s path as can be seen in Figure 3.9 and distance Table 3.4 for the Levenshtein logic. In Figure 3.10 and distance Table 3.5, the Damerau-Levenshtein path through the graph and associated distance matrix is displayed. In Table 3.6, the associated tuples $P$ and related operations are displayed in a summary format.
Figure 3.9: Levenshtein Traced-Back Shortest Path Running Example

Table 3.4: Levenshtein Shortest Path Through Running Example Distance Matrix $D$

<table>
<thead>
<tr>
<th></th>
<th>New York</th>
<th>Philadelphia</th>
<th>Bel Air</th>
<th>Baltimore</th>
<th>Washington</th>
<th>Salisbury</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Philadelphia</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Bel Air</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Baltimore</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Washington</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
Table 3.5: Damerau-Levenshtein Shortest Path Through Running Example Distance Matrix

<table>
<thead>
<tr>
<th></th>
<th>Phila-</th>
<th>Bel</th>
<th>Baltimore</th>
<th>Washington</th>
<th>Salisbury</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>New York</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Philadelphia</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Washington</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Baltimore</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3.6: (Damerau-)Levenshtein Operations Summary Running Example $x$ to $y$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$P_k^1$</th>
<th>$P_k^2$</th>
<th>$O_k$</th>
<th>$P_k^1$</th>
<th>$P_k^2$</th>
<th>$O_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Delete</td>
<td>1</td>
<td>1</td>
<td>Substitute</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>Match</td>
<td>2</td>
<td>2</td>
<td>Substitute</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>Substitute</td>
<td>3</td>
<td>3</td>
<td>Transpose</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>Match</td>
<td>4</td>
<td>4</td>
<td>Transpose</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>4</td>
<td>Insert</td>
<td>4</td>
<td>5</td>
<td>Insert</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>5</td>
<td>Insert</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The explicit edit operations that are displayed in Table 3.6 are displayed graphically in Figures 3.11 and 3.12 for the Levenshtein and Damerau-Levenshtein based logic respectively. The figures show the explicit edit operations to obtain activity sequence $y$ from activity sequence $x$, where only the edit operations associated with the identification similarity are incurred. The arrows connect the activities that are associated with the individual edit operations.

Figure 3.11: Levenshtein Based Operations Running Example Graphical
At this point, the identification similarity edit operations to obtain sequence $y$ from sequence $x$ have been defined. More specifically, the most efficient application of the different edit operations have been made explicit. This involves that it is now clear how, where, how often and with which associated customers activities the difference between two sequences can be described in terms of identification similarity edit operations. The identified match and transpose edit operations can be used to describe structural similarities one step further by including temporal differences. The temporal similarity will be further elaborated in the following section.

3.2 Temporal Similarity

In the previous section, the process of retrieving identification similarity edit operations to transform one sequence of activities into another has been explained. The following section proposes a temporal similarity assessment of the activities associated with the identification similarity edit operations match and transpose, which are related to corresponding activities. The temporal similarity assessment of these corresponding activities is done by calculating temporal differences in terms of start time and duration. First, the different edit operations are placed into three groups based on their nature. These groups involve:

1. Matched (a pair of identical activity identifications in both sequences)
2. Transposed (two pairs of transposed identical activity identifications)
3. Deleted, inserted and substituted (non-identical activity identifications)

For the edit operations related to identical items(customers/activities), matched and transposed items, the comparison is enhanced by including ratios for the duration and difference in start time. It has been chosen to include both measures to allow examined activities to be sequential but not to start directly after the preceding activity. All edit operation groups shall be multiplied with a ratio of their level of occurrence relative to the total number of edit operations. Therefore, this ratio will be referred to as the occurrence ratio.

**Matched and transposed identification similarity edit operations** When an edit operation match (m) is found, this means that an activity or customer is found in the same place in the sequence order of both examined sequences. It may also be true that after some other operations such as insert, delete or substitute a match is found. For an operation of
group transposed (t), two adjacent customers or activities are found in both strings but in different order.

The temporal assessment between two sequences of factors in the groups matched and transposed can be summarized in the following similarity score factors:

1. Difference in Start Time (∆ST)
2. Difference in Duration Time (∆DT)

The difference in start time ∆ST between any operation type at position \(i\) in sequence \(x\) and at position \(j\) in sequence \(y\) is equal to \(|ST_{i,x} - ST_{j,y}|\). The difference in duration time ∆DT between any operation type position \(i\) in sequence \(x\) and position \(j\) in sequence \(y\) is equal to \(|ST_{i,x} - ET_{i,x}| - |ST_{j,y} - ET_{j,y}|\). This attribute was described by Wall (1996) as the Euclidean distance measure. To obtain ratios to what extent these similarity score factors contribute to the similarity score, the scores are divided by the length of a uniform length of time. This ratio has been chosen over a ratio that is related to a pair of schedules because in comparing different schedules with one target schedule, a common denominator should be in place to function as a comparative constant. In the current research, the time length of 24 hours is adopted. The cost factor of all shifts for all matches between schedule \(x\) and \(y\) is constructed by averaging the shift differences in all operations of type \(m\) and \(t\) respectively.

The similarity score factor average Difference in Start Time \(\frac{\Delta ST}{M}\) and the similarity score factor average Difference in Duration Time \(\frac{\Delta DT}{M}\) are thus defined as follows for the groups matched and transposed respectively.

\[
\Delta ST_M = \frac{1}{|M|} \sum_{(i,j) \in M} \frac{|ST_{i,x} - ST_{j,y}|}{24}
\]
\[
\Delta DT_M = \frac{1}{|M|} \sum_{(i,j) \in M} \frac{|ST_{i,x} - ET_{i,x} - |ST_{j,y} - ET_{j,y}|}{24}
\]
\[
\Delta ST_T = \frac{1}{|T|} \sum_{(i,j) \in T} \frac{|ST_{i,x} - ST_{j,y}|}{24}
\]
\[
\Delta DT_T = \frac{1}{|T|} \sum_{(i,j) \in T} \frac{|ST_{i,x} - ET_{i,x} - |ST_{j,y} - ET_{j,y}|}{24}
\]

The measurement is now explained following the running example. As was shown in Table 3.6, following the Levenshtein logic, two matches were found. Namely in step 3 at \(P_1^3, P_2^3\) or \(D(2,1)\) \(x_2\) and \(y_1\): "Philadelphia" and in step 5 at \(P_5^4, P_6^4\) or \(D(4,3)\) \(x_4\) and \(y_3\): "Baltimore". In Table 3.7, the calculation is shown to achieve similarity score factors \(\Delta ST\) and \(\Delta DT\).
As is shown in Table 3.6, when using the Damerau-Levenshtein similarity assessment method, a pair of transposed locations was found. Namely, in step 4 and 5 with \( x[3, 4] \) and \( y[4, 3] \): "Baltimore" and "Washington D.C.". In Table 3.8, the calculation is shown to achieve similarity score factors \( \Delta ST_T \) & \( \Delta DT_T \).

<table>
<thead>
<tr>
<th></th>
<th>Washington D.C.</th>
<th>Baltimore</th>
<th>( \Delta ST_T )</th>
<th>( \Delta DT_T )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_k^1 )</td>
<td>3</td>
<td>4</td>
<td>0.120</td>
<td>0.005</td>
</tr>
<tr>
<td>Start time</td>
<td>13:15</td>
<td>15:10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>End time</td>
<td>14:15</td>
<td>15:30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( P_k^2 )</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start time</td>
<td>12:00</td>
<td>10:40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>End time</td>
<td>13:00</td>
<td>10:55</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Delete, insert and substitute identification similarity edit operations Presence of the operations delete(d), insert(i) and substitute(s) indicate complete difference between the pair of examined schedules in terms of identification similarity. Therefore, the temporal properties of these operations are not evaluated as these would not make sense. Therefore, the score that is related to these operations and contributes to the final distance score is equal to 1 for each of these operations, which is the maximum distance. The joint contribution is referred to as the similarity score factor \( DIS \).

The DIS similarity score factor will now be assessed with respect to the running example. As is shown in Table 3.6, the \(|D|, |I|, |S|\) operations equal 1, 2, 1 respectively for the Levenshtein operations. Therefore, the combined contribution \( DIS \) is the sum of the elements, which is 4 in total. Similarly, the values for the Damerau-Levenshtein operations equal 0, 2, 1 respectively. Therefore, the combined contribution to this factor is 3.
3.3 The Proposed Metric

In the previous sections, focal characteristics of route planning and route optimization have been discussed. Additionally, different comparison approaches have been treated. In the following section, the approaches will be synthesized into one metric, the proposed metric. A conceptual overview of the metric is displayed in Figure 2.2. As was shown in the previous section, the Damerau extension is analyzed separately. For the proposed metric, it suffices to present one version, because when the Damerau extension is not taken into account, the associated parameters equal to zero. The proposed metric is composed of a two-steps measurement. In the first step, the operations required to efficiently change the identification aspects of one sequence of activities into another sequence are uncovered. These operations include: match(m), transpose(t), delete(d), insert(i) and substitute(s). Now, it holds that the number of operations $|O|$ equals the number of matches $|M|$ plus the number of transpositions $|T|$ plus the number of deletions $|D|$ plus the number of insertions $|I|$ plus the number of substitutions $|S|$.

Therefore, the following edit operations occurrence ratios add up to 1:

$$\frac{|M|}{O} + \frac{|T|}{O} + \frac{|D|}{O} + \frac{|I|}{O} + \frac{|S|}{O} = 1$$

For the Levenshtein running example, this means:

$$\frac{2}{6} + \frac{0}{6} + \frac{1}{6} + \frac{2}{6} + \frac{1}{6} = 1$$

The Damerau-Levenshtein running example leads to the following distribution of the edit operations occurrence ratios:

$$\frac{0}{5} + \frac{2}{5} + \frac{0}{5} + \frac{1}{5} + \frac{2}{5} = 1$$

In the second step of the proposed metric, the temporal factors for matched and transposed elements are added to the final equation of the proposed similarity score.

$$proposedScore = 1 - \left( \frac{|M|}{O} * (A * \Delta ST_M + B * \Delta DT_M) + \frac{|T|}{O} * (C * \Delta ST_T + D * \Delta DT_T) + E * (\frac{|D| + |I| + |S|}{|O|}) \right)$$

Where $A, B, C, D, E$ are parameters indicating the relative, trainable importance of the different operations and $\sum A + B + C + D + E = 1$.

The final proposed similarity score to obtain running example routing schedule $y$ from $x$ based on the Levenshtein logic thus results in:

$$proposedScore = 1 - \left( \frac{2}{6} * (A * 0.139 + B * 0.10) + \frac{0}{6} * (C * 0 + D * 0) + E * (\frac{1}{6} + \frac{2}{6} + \frac{1}{6}) \right)$$
The final proposed similarity score to obtain running example routing schedule $y$ from $x$ based on the Damerau-Levenshtein logic thus results in:

$$proposedScore = 1 - \left( \frac{|0|}{5} \right) \ast (A \ast 0 + B \ast 0 + \frac{|2|}{5} \ast (C \ast 0.120 + D \ast 0.005) + E \ast \left( \frac{|0| + |1| + |2|}{5} \right))$$
Chapter 4

Results

In this chapter, the parameters of the proposed method are trained in the context of truck routing schedules to test its usefulness and applicability in this area. The proposed method is constructed in such way that it can be trained according to the dataset or application area it is used for. In the first section, the training procedure will be treated. In the second section, the data selection and scenario description will be covered. This will lead to the presentation of the experiment design and participant selection in section three. The chapter will close with a presentation of the assessment results in section four.

4.1 Training Procedure

In this section, the procedure of training the proposed metric is set forth. Experts are asked to rank candidate routing schedules on their similarity with target routes. Their input is used to train parameters set $A - E$ of the proposed method to mimic the expert judgment as closely as possible. First, the accuracy of an estimate is explained. Second, the formal training procedure to train the assessed scenarios is treated.

4.1.1 Accuracy estimation.

The accuracy of an assessed set of parameters is defined as the ratio of difference in the ranking between the expert input similarity ranking and the ranking provided by the proposed metric, compared to the maximum difference in ranking. This determination is similar to the most common accuracy determination of classification algorithms, which is defined as the classification rate, or the number of successful hits relative to the total number of classifications (Garcia et al., 2009; Kotsiantis, 2007). Consider target route $i$ with routes $j$ to be ranked on their respective similarity. The ranking is done by the proposed metric $PM$ and the expert input $EX$. The similarity ranking $R$ of route $j$ with respect to target route $i$ by the proposed metric $PM$ and the expert input $EX$ is thus defined as $R_{PM,i,j}$ and $R_{EX,i,j}$ respectively. Then, the absolute difference between the two similarity rankings for target route $i$ can be defined as:

$$\sum_j |R_{PM,i,j} - R_{EX,i,j}|$$
The accuracy of an $R_{PM}$ is defined as one minus the ratio of the absolute difference between the two similarity rankings divided by the maximum difference in ranking. The maximum difference between two separate rankings is reached when the rankings are the complete opposite of each other. In this research, five ranks are divided. As can be seen in Table 4.1, the maximum difference between any two ranking of five rankees is thus equal to $4 + 2 + 0 + 2 + 4 = 12$.

Table 4.1: Maximum Difference Determination Between any two Rankings of Five Rankees

<table>
<thead>
<tr>
<th>Ranking A</th>
<th>Ranking B</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

Consequently, the accuracy of an estimation by the proposed metric is defined as:

$$1 - \frac{\sum_{j} |R_{PM,i,j} - R_{EX,i,j}|}{12}$$

### 4.1.2 Model training.

In this section, the procedure to train the proposed metric is explained. The training is carried out in a two step method.

1. Calculate for the selected sets of routing schedules the separate similarity score factors that the proposed metric is composed of. These factors include the occurrence ratios of edit operations and the average temporal differences in duration and starting time.

2. Maximize accuracy of proposed metric ranking by changing parameters A-E between 0 - 1 by incrementing steps of 0.01.

The minimization will be done according to the following logic: Consider a total of $i$ exercises consisting of $j$ rankee routes. These positions mean that if route $j$ in exercise $i$ is ranked on $R_{PM,i,j} = 1$, it shows most similarity with the target route in exercise $i$. Further, calculate the difference between proposed metric rank $R_{PM}$ and expert similarity rank $R_{EX}$ of route pair $(i, j)$ by altering parameters $A - E$ in the calculation of the proposed metric score such that $\sum A + B + C + D + E = 1$ and the proposed score representing a similarity value between 0 - 1 with 1 showing the highest possible similarity and 0 the lowest. Last, divide the resulting difference by the maximum difference to achieve the average accuracy of a set of parameters. This results in the following objective function for training the model:

$$\max 1 - \frac{\sum_{i} \sum_{j} |R_{PM,i,j} - R_{EX,i,j}|}{12}$$

### 4.2 Data Selection and Scenario Description

In the following section, the data that is used in the different assessed scenarios is described. Thereafter, the different assessed scenarios are set forth.
4.2.1 Data selection.

The data that is used in this study contains all routing schedules of a leading Dutch truck transportation company that have been carried out from January 1st, 2016 to April 22nd, 2016. The set contains 130999 stops across 15185 addresses where many locations are visited only once. The proposed metric has a preference for elements across the two assessed sequences that are identified as a match. Therefore, the route selection is important with respect to this attribute. Huang et al. (2009) indicate that visually comparing even two schedule records is experienced as tedious and time-consuming. Therefore, to improve judgment, quality and reliability, the routes that are compared should not involve too many stops. In this study, data is selected according to the following procedure.

1. Create a subset of routes between 8 and 12 stops to improve experts’ ease of comparison.
2. Randomly select 15 routes that relate to five or more routes with a match operations ratio larger than zero. These are the target routes.
3. For each target route, randomly select five routes out of the routes that have a match operations ratio larger than zero. These are the rankees.

4.2.2 Assessed scenarios.

In this subsection, the scenarios that are investigated with the proposed metric in the area of truck routing schedules are discussed. In one stream of analysis, the proposed metric will be trained and tested using the results from expert input. In the other stream of analysis, the same procedure is done with random results, or monkey expert input, for the training and validation of the proposed method. This approach has been chosen to test the added value of both the proposed metric and the questionnaire method used in this study. These two training streams will be tested following the two discussed algorithms in order to determine the identification similarity edit operations. The original Levenshtein algorithm will be used and compared with the Damerau-Levenshtein. To compare between the monkey and expert input, the same route combinations are used. However, in the monkey scenario, the expert judgment is randomized. The dataset is split into a 70% training set and a 30% test set. An overview of the assessed scenarios is displayed in Table 4.2.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Qualitative logic</th>
<th>Training set</th>
<th>Test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Levenshtein</td>
<td>70% of expert set</td>
<td>30% of expert set</td>
</tr>
<tr>
<td>2</td>
<td>Levenshtein</td>
<td>70% of monkey set</td>
<td>30% of monkey set</td>
</tr>
<tr>
<td>3</td>
<td>Damerau-Levenshtein</td>
<td>70% of expert set</td>
<td>30% of expert set</td>
</tr>
<tr>
<td>4</td>
<td>Damerau-Levenshtein</td>
<td>70% of monkey set</td>
<td>30% of monkey set</td>
</tr>
</tbody>
</table>

4.3 Experiment Design

In the following section, the experiment design to validate the proposed metric in comparing truck routing schedules is discussed. The metric is validated by training and testing the
logic with (monkey) expert input. In the first section, the questionnaire design is described. In the second section, the participant selection is treated.

4.3.1 Questionnaire design.

Each questionnaire consists of fifteen truck route similarity ranking exercises. Further, each separate exercise consists of a visual representation of six truck routing schedules. The participant is asked to rank five routing schedules according to their similarity with a target routing. The instructions to do the ranking are displayed in Appendix A. Harjunkoski et al. (2014) treated some methods to visually display planning schedules in different context. In this research, their proposed concept of displaying precedence-based schedules in a single, common grid is adopted. According to Psaraftis (1988), in a static scheduling problem or solution, times are assumed proportional to the distances traveled. This implies that a schedule can be described either using time or distance. Therefore, in the representation, the travel time or distance is displayed as a blank area between visited customers. The construction references of Psaraftis (1988); Harjunkoski et al. (2014) led to a representation that is made according to the following steps.

1. For each set of target route and runkees, assign colors and anonymized addresses to unique customers
2. Display the routes in a single, common grid with blank spaces between the serviced customers representing travel time

An example of a questionnaire exercise is displayed in Figure 4.1.

![Figure 4.1: Example Questionnaire Exercise](image)

4.3.2 Participant selection.

The participants that are required to train the proposed metric are approached through the Accenture network. Accenture is a global leading professional service provider that works with over 75% of the Fortune 500 companies. The industries they have expertise in range from banking to health and utilities. The services are provided not only on a strategic level, but also on technological or operational level (Accenture LLP, 2017). In this research, the scope of the proposed metric is focused on routing schedule planning in the trucking sector.
Therefore, experts are involved that have proven experience in logistics and supply chain planning. The records of the seven participants that have participated in this research can be found in Appendix B.

4.4 Assessment Results

In this section, results of testing the proposed similarity measurement method in the context of comparing trucking routing schedules will be treated. The section is subdivided into three subsections. In the first section, an overview of the similarity factors across the assessed sets of routing schedules will be shown. Here, the parameters are not yet incorporated. The second section will treat the parameter training results of the proposed similarity measurement method across the different assessed scenarios. These scenarios include (monkey) expert training of the Levenshtein and Damerau-Levenshtein based proposed metric. In the assessment of the results, the accuracy of the proposed method in replicating the expert input is investigated.

4.4.1 Similarity factors across assessed routing schedule groups.

In Table 4.3, a summary of the assessed pairs of routing schedules and their calculated factor scores are displayed. The DIS ratio is defined as the sum of delete, insert and substitute operations, divided by the total edit operations. In the analysis, the input of seven experts is taken into account. Each expert is required to fill out 15 ranking exercises. Each exercise consists of ranking five routes according to their similarity with the target routes. Therefore, the total amount of pairs that are assessed on their similarity score yields $5 \times 15 \times 7 = 525$ pairs.

<table>
<thead>
<tr>
<th>Similarity factor</th>
<th>Levenshtein</th>
<th>Damerau-Levenshtein</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio of DIS elements</td>
<td>$\mu$ 78.73%</td>
<td>$\mu$ 77.59%</td>
</tr>
<tr>
<td></td>
<td>$\sigma$ 14.18%</td>
<td>$\sigma$ 15.44%</td>
</tr>
<tr>
<td>Ratio of matched elements</td>
<td>$\mu$ 21.27%</td>
<td>$\mu$ 20.44%</td>
</tr>
<tr>
<td></td>
<td>$\sigma$ 14.18%</td>
<td>$\sigma$ 14.19%</td>
</tr>
<tr>
<td>Duration difference matched elements (hours)</td>
<td>0.152</td>
<td>0.153</td>
</tr>
<tr>
<td></td>
<td>0.156</td>
<td>0.148</td>
</tr>
<tr>
<td>Shift difference matched elements (hours)</td>
<td>0.792</td>
<td>0.784</td>
</tr>
<tr>
<td></td>
<td>1.107</td>
<td>1.111</td>
</tr>
<tr>
<td>Ratio of transposed elements</td>
<td>0%</td>
<td>1.97%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6.00%</td>
</tr>
<tr>
<td>Duration difference transposed elements</td>
<td>/</td>
<td>0.202</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.186</td>
</tr>
<tr>
<td>Shift difference transposed elements (hours)</td>
<td>/</td>
<td>0.673</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.672</td>
</tr>
</tbody>
</table>

4.4.2 Assessed scenarios results.

In the following section, the results of the assessed scenarios will be treated. The data that is used for the training consists of 70% randomly selected groups of routing schedules. The test set contains the other 30% of routing schedule groups. In Table 4.4, the training and testing results of the different assessed scenarios are displayed. The displayed parameters
are the parameters that have shown the best results in the training of the expert and monkey input respectively. In the row Test accuracy, the accuracy of the similarity measurements in the test set is displayed.

In the proposed metric based on the Levenshtein logic, the best performing parameter set yields a training accuracy of 43.52% for the expert training, and a 9.2% higher training accuracy of 47.52% when the model is trained by random rankings. The test accuracy of the Levenshtein based metric is higher when trained by random ranking than when trained by experts. The expert test accuracy is 23.5% lower than the monkey results, 32.29% and 42.19% respectively.

In the proposed metric based on the Damerau-Levenshtein logic, the best performing parameter set yields a training accuracy of 42.69% for the expert training, and a slightly higher accuracy for the monkey training (43.38%). The expert test accuracy is 11.9% higher than the monkey results, 39.06% and 34.90% respectively.

In comparison between both metric bases, the metric shows the highest overall test accuracy when trained by random rankings and when the metric is based on the Levenshtein logic. In both logics, monkey training yields a higher training accuracy than expert training does. The Damerau-Levenshtein based assessment showed better expert test accuracy than the Levenshtein based assessment.

Table 4.4: Training and Test Results

<table>
<thead>
<tr>
<th>Metric Basis</th>
<th>Levenshtein</th>
<th>Damerau-Levenshtein</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>Expert</td>
<td>Monkey</td>
</tr>
<tr>
<td>A</td>
<td>0.11</td>
<td>0.19</td>
</tr>
<tr>
<td>B</td>
<td>0.57</td>
<td>0.25</td>
</tr>
<tr>
<td>C</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>D</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>E</td>
<td>0.32</td>
<td>0.56</td>
</tr>
<tr>
<td>Training accuracy</td>
<td>43.52%</td>
<td>47.52%</td>
</tr>
<tr>
<td>Test accuracy</td>
<td>32.29%</td>
<td>42.19%</td>
</tr>
</tbody>
</table>

In all, the accuracy of the proposed metric in the investigated comparison of truck routing schedules may be considered low. Additionally, albeit the expert training showed better results than the monkey training in the Damerau-Levenshtein based metric, the accuracy may still be considered poor in this application area.

Moreover, the proposed metric has a preference for matched locations. As can be seen in Table 4.3, the average ratio of matched items in investigated pairs of routes lies below 22%. It may be true that the low average ratio may results in a low test accuracy for two reasons. First of all, the training accuracy of the proposed metric may suffer from the low amount of matches as the temporal section is a more detailed investigation of the matched elements. When the number of matches is low, the metric can compare on less elements. Second of all, the low relative number of matches in the dataset may make the expert judgment call less reliable, leading to a lower quality of the expert input. It is hypothesized that a larger ratio of match operations in a similarity assessment leads to an increased quality in expert judgment and detail assessment among the matched elements. This hypothesis is tested by re-assessing route sets that have been judged by the involved experts and have a relatively high average in-group match ratio. A training accuracy achieved in this subset higher than the accuracy achieved in assessing all investigated sets of routes would support
the hypothesis. The investigated set of questionnaires is divided into three groups of match ratio: 20%-25%, 25%-30% and above 30%. The resulting parameters and training accuracy of the proposed metric based on the Levenshtein logic is displayed in Table 4.5. The results show a slightly increased accuracy when the match ratio increases. Conclusively, although the accuracy differences between each group of match ratios are small, as is the number of investigated questionnaire sets, the data shows some support towards the hypothesis.

Table 4.5: Training Accuracy Across Different Match Ratio Subsets Based on Levenshtein Logic

<table>
<thead>
<tr>
<th>Match ratio</th>
<th>20%-25%</th>
<th>25%-30%</th>
<th>30%+</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.17</td>
<td>0.09</td>
<td>0.40</td>
</tr>
<tr>
<td>B</td>
<td>0.83</td>
<td>0.91</td>
<td>0.52</td>
</tr>
<tr>
<td>C</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>D</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0.08</td>
</tr>
<tr>
<td>n</td>
<td>22</td>
<td>19</td>
<td>12</td>
</tr>
<tr>
<td>Training accuracy</td>
<td>40.15%</td>
<td>42.11%</td>
<td>43.06%</td>
</tr>
</tbody>
</table>
Chapter 5

Relevance

In the following section, potential use of the treated similarity logic in a business context is discussed. The proposed metric makes use of both identification similarity edit operations and temporal similarity factors to determine structural similarity among activity sequences. One of the features of the method is that it allows to present the steps to obtain one sequence of activities from another in an explicit form. More specifically, the metric may be used as an efficient search method to find suggestions to incrementally or radically improve an assessed sequence of activities.

In the following section, a potential business application of the proposed metric is explored. The truck transportation data that has been used throughout this study has shown a relatively low accuracy with respect to similarity determination by the proposed metric. Therefore, the applications are described conceptually. First, a similarity-based decision support tool based on historical data is displayed. This tool may support in achieving incremental improvements of activity sequences. Second, a machine learning extension on this decision support tool is provided. This extension may support in achieving radical improvements of activity sequences. The decision support tool will be illustrated following a similar running example that was used to explain the proposed metric. In the first section, the conceptual tool will be treated and illustrated with a prototypical example in the second part. In the third section, the machine learning extension will be covered.

5.1 Conceptual Similarity-Based Decision Support Tool (SDSS)

The proposed similarity-based decision support tool (SDSS) leverages the structural comparison properties of the proposed metric in this study in a constructive, improvement-oriented way. The proposed metric has a descriptive nature; in that sense, the quality evaluation of an activity sequence may also be regarded in a descriptive manner. This evaluation may differ from one area to another but can possibly be represented by a quality evaluation score in any area. Examples of such evaluation scores may be timeliness, cost, profit or reliability. Moreover, as using only one of such evaluation measures seldom suffices a good evaluation, a combination of different scores may be used. In a generic form, activity sequences that may be incorporated into the SDSS take a form such as displayed in Figure 5.1.
This activity sequence representation combined with the proposed metric can be used in different supporting ways. One of the supporting ways is by incrementally improving candidate solutions based on their similarity, quality evaluation and boundary constraints. Here, an example of such supporting mechanism is displayed.

1. Feed candidate activity sequence
2. Find an activity sequence in the sequence database that has the highest similarity score
3. If the quality evaluation of the suggested sequence is better than the candidate’s and is allowed within boundary constraints, the suggested sequence becomes the candidate sequence
4. Assess next similar activity sequence until convergence is achieved
5. Explicitly map the operations to obtain the suggested activity sequence in terms of identification similarity edit operations and temporal difference

### 5.2 Prototypical Example

As discussed earlier, the quality evaluation measure can be selected based on the domain in which the SDSS is implemented. In this thesis, the proposed metric is guided by an example in the transportation sector. Gunasekaran et al. (2004) presented a framework for supply chain performance measurement. In the context of delivery performance, \textit{on time delivery of goods}, \textit{quality of delivered goods} and \textit{flexibility of service systems to meet customer
needs are classified as highly important measures. _On time delivery_ has also been identified by Solomon (1987) to be an important performance measure in vehicle routing context. Therefore, the _on time delivery_ of goods is adopted as the quality evaluation measure in the prototypical example elaboration of the SDSS. In the example, the total minutes of deviation in the start of service time that was pre-agreed on by the delivery party and the customers is used as the quality evaluation score in the prototypical example. This measure is referred to as _tardiness_.

Consider the routes and tardiness values that will form the database of the running example in Table 5.1. In Table 5.2, the prototypical similarity scores among the four routes is displayed.

<table>
<thead>
<tr>
<th>Route</th>
<th>Stop</th>
<th>Start time</th>
<th>Duration</th>
<th>Tardiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>New York</td>
<td>08:00</td>
<td>00:20</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Philadelphia</td>
<td>10:10</td>
<td>00:15</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Washington</td>
<td>13:15</td>
<td>01:00</td>
<td>12</td>
</tr>
<tr>
<td>B</td>
<td>New York</td>
<td>14:00</td>
<td>00:20</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Philadelphia</td>
<td>16:10</td>
<td>00:15</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Washington</td>
<td>19:15</td>
<td>01:00</td>
<td>27</td>
</tr>
<tr>
<td>C</td>
<td>New York</td>
<td>11:00</td>
<td>00:20</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Washington</td>
<td>14:00</td>
<td>00:25</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>Philadelphia</td>
<td>16:00</td>
<td>01:00</td>
<td>6</td>
</tr>
<tr>
<td>D</td>
<td>New York</td>
<td>13:00</td>
<td>00:20</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Philadelphia</td>
<td>15:30</td>
<td>00:25</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 5.1: Prototypical Route Database
In the accompanying example, the boundary condition holds that New York, Washington and Philadelphia should be visited in one route. Furthermore, tardiness is the sole quality measure that is taken into account. Route A will function as the starting candidate route. The proposed SDSS would first assess the historical route that has the highest similarity. Route B has the highest similarity score with similarity score (A,B) is 0.94, as can be seen in Table 5.2. In Table 5.1, Route B seems to differ on its structure only on the time that the route has been carried out, namely six hours later. Therefore, the truck encountered a severe prototypical traffic jam which increased the tardiness. Route B has a tardiness of 61 minutes, which is higher than the candidate route. Therefore, the route is not better than the candidate and thus does not become the new candidate route. Now, the SDSS further investigates the second most similar route retrieved from Table 5.2. With a similarity score (A,C) of 0.92, route C is the potential next candidate route. When reviewing the route database, we see that the quality assessment is better than route A and the boundary condition is met. Therefore, route C becomes the next candidate solution. Now, the SDSS will assess the other routes. The only route that has a lower tardiness than route C is route D, but it violates the set boundary condition. Based on the SDSS, the best improvement suggestion is to change route A to route C. The change can be described in the following two steps:

1. **Identification similarity edit operations** First, the identification similarity logic is applied to obtain the next sequence. In case of the Levenshtein logic, this would involve
behind the scenes: Remove Philadelphia at position two, insert Philadelphia at position three. For the Damerau-Levenshtein, this would imply a transposition between Philadelphia and Washington.

2. Temporal change Second, the temporal logic of the proposed metric is applied to the identical elements, which implies a shift of all visited customers along the time line and a slight change in the service time. A graphical representation of the steps in the SDSS is displayed in Figure 5.3.

![Figure 5.3: Graphical Representation of SDSS Proposed Improvement Suggestion](image)

5.3 Machine Learning SDSS Extension

In the next section, a proposed mechanism is explained that extends the discussed SDSS with a machine learning property allowing to create new activity sequences. In the ordinary SDSS, the database of evaluable activity sequences is filled with historically executed activity sequences. The proposition of the machine learning extension is to append the database with new activity sequences that are composed based on feasibility rules learned from the historical activity sequences.

A high-level overview of a conceptual implementation of the total proposed ecosystem that includes both the SDSS and the machine learning extension is displayed in Figure 5.4. Step 1 involves the retrieval of historical planning schedules. In the conceptual implementation, it is assumed that these schedules have been constructed by experts. The historical schedules are stored in the repository of feasible schedules. This repository would have a central function in the implementation. Now, problem solving and new schedule improvements can be executed by interacting with this repository following execution processes A and B. Step 1 and execution processes A and B form the proposed SDSS.

In Step 2, Learn, an overview is provided of the role that a machine learning extension could fulfill in this ecosystem. First of all, the schedule feasibility rules are extracted from the historical schedules. These feasibility rules can for instance consist of resource capacity, time constraints or sequence order rules. At this point, the situation can be created to assess
new activity sequences on their feasibility in a simulation environment by incorporating the extracted rules.

The deduction of feasibility rules and common properties can for instance be done by feature selection using genetic programming. Genetic programming algorithms search within a topological space, allowing it to define structures and mathematical expressions (Li et al., 2006). Moreover, Kinnear et al. (1999) found that genetic programming was in 1999 already at a state that it is able to produce many results that are competitive with human-produced results. Koza (2008) even explains that genetic programming now routinely delivers human-competitive machine intelligence for problems of automated design and can even serve as an automated invention machine. Through operations such as crossover and mutation among the examined activity sequences, better solutions can be created that would not be obtained with incremental improvement. This evolutionary process of solutions generation may remove human negative influence and bias and helps to conduct feature extraction from a much larger space (Guo et al., 2005). Conclusively, the incorporation of genetic programming in the machine learning SDSS ecosystem may reduce human bias, increase the extent of extracted features and automate activity sequence invention with human-competitive intelligence.

Figure 5.4: SDSS and Machine Learning Ecosystem
Chapter 6

Conclusion

In this section, the objectives of this thesis will be evaluated. The primary objective of this research is to develop a metric that allows to assess structural similarity among generic sequences of activities. The secondary objective is to test if truck routing schedules can be described and compared based on structural similarity and a descriptive point of view. The tertiary objective is to explore the applicability of the metric in a decision support role for improving sequences of activities. First, the proposed metric is discussed. Second, the added value of this thesis’ products is covered. In the third section, the assessment of the proposed metric in the truck routing domain will be covered. In the fourth part of this chapter, the business relevance of this research will be treated. In the last part, limitations and directions for future work with respect to this thesis’ products are covered.

6.1 The Proposed Metric

The proposed metric allows to measure structural similarity among generic sequences of activities from a descriptive point of view. The (Damerau-)Levenshtein text similarity metrics have been used to compare the unique identifiers of activities. By regarding unique identifiers as (sub-)strings, the text similarity metrics have been leveraged to extract explicit edit operations how to obtain one sequence of identifiers from another sequence. The proposed metric has been constructed in such way that it produces a similarity value between zero and one. Further, by training the incorporated parameters, the importance of each individual component can be adjusted to the domain that the proposed metric is trained in. When two sequences are identical and thus show only matches with a temporal difference of 0, the resulting similarity score is equal to one, regardless of the training outcome. Moreover, the remaining parameters can be trained in such a way that the similarity score decreases according to the priority of the other measured similarity factors, depending on the domain that the metric is trained in. Therefore, it is concluded that the primary objective of this research has been met.

6.2 Scientific Added Value

The proposed metric to obtain a similarity score between two sequences of activities contains elements that can be regarded as scientific added value. First of all, the Levenshtein and
Damerau-Levenshtein metrics have been extended by leveraging the logic to extract explicit edit operations. These edit operations stem from the original logic and include match, delete, insert, substitute and transpose where the last is only incurred in the Damerau variant. These explicit operations are retrieved by first finding a shortest path through the distance matrix using Dijkstra’s algorithm and then to re-apply the logic that was used to construct the distance matrix. The logic has been even further extended by recognizing insert-match and delete-match patterns from the distance matrix. It may also be an added value that it has been shown that complete strings of characters can be compared with these logics instead of individual characters. Besides, a similarity search research direction is proposed where an assessment of a temporal level is added to the comparison of sequences of uniquely identifiable sequential items.

6.3 Suitability of Proposed Metric in Truck Route Comparison

In this section, the assessment of the proposed metric in the truck routing schedule comparison domain is treated. In chapter 2, a literature research has been performed on the composition and on the comparative improvement heuristics of truck routing schedules. The investigation led to the conjectures that a truck routing schedule can be described as a generic sequence of activities and that descriptively comparing routing schedules may yield better than random results. Based on this research, there has not been found evidence to refute nor support Hypothesis 1 that a routing schedule can be described as a generic sequence of activities.

Furthermore, the accuracy of the proposed metric in the domain of truck routing schedules has shown a poor accuracy. This may be due to the suitability of the domain and the proposed metric, the used data or the evaluation of the accuracy. Therefore, objective two is achieved but it may be concluded that the domain truck routing schedule comparison is not suitable for the proposed metric. The Levenshtein based metric showed a worse than random ranking accuracy and the Damerau-Levenshtein based metric showed a better than random ranking accuracy. However, all metric bases were outperformed by the Levenshtein based random ranking accuracy, therefore hypothesis 2 is rejected.

6.4 Relevance

In chapter five, the relevance of the proposed metric has been explored conceptually in business context. The proposed metric may serve as an efficient method to generate improvement suggestions. By assessing improvement possibilities that are within a pre-determined distance from the candidate activity sequence and have a better quality evaluation, explicit edit operations can potentially be achieved fairly quickly. A trade-off needs to be considered when determining how many boundary conditions are adopted in the evaluation part. A large number of boundary conditions would lead to less but possibly more feasible solutions. The potential of the proposed assisted problem solving and improvement suggestion may increase the speed of problem solving and improve the quality of activity sequences. Moreover, the suggested SDSS machine learning extension can advise on or invent new solutions, that would potentially require much expert experience to spot. Furthermore, the extent of feature extraction can be increased, which could lead to an outperforming of
human-produced solutions. The SDSS and machine learning extension may be leveraged by organizations for a variety of beneficiary purposes. An implementation of the proposed SDSS could potentially lead to the following benefits:

- Faster problem solving through similarity search in comparable problems
- Incremental activity sequence improvement through search in comparable sequences
- Concrete roadmap from current to new activity sequence

The genetic programming extension could enhance the SDSS leading to the following additional benefits:

- Radical activity sequence improvement through reduced human bias and an increased extent of feature selection
- Radically new activity sequence generation through automated activity sequence invention

6.5 Limitations and Suggestions for Further Research

In this section, suggestions for further research are discussed. The section is structured according to the chronologically delivered products in this thesis. These products include: 1) the proposed extension of the (Damerau-)Levenshtein logic to explicitly define edit operations, 2) the proposed metric to compare two generic activity sequences, 3) the descriptive similarity assessment of truck routing schedules and 4) the conceptual descriptions of business relevant applications in similarity-based activity sequence improvement.

6.5.1 The proposed extension of (Damerau-)Levenshtein logic.

In this research, an extension has been proposed on the (Damerau-)Levenshtein logic that enables the extraction of explicit edit operations between two strings, or sequences of strings, based on first finding a shortest path through the distance matrix resulting from the edit distance logic and then to back-trace the explicit edit operations associated with a shortest path that has been found with Dijkstra’s algorithm. Naturally, different paths may lead to the same outcome. Furthermore, for a metric to be a valid similarity metric, the following conditions should be satisfied for activity sequence pair \((x,y)\):

1. Non-negativity of separation axioms: \(sim(x,y) \geq 0\)
2. Identity of indiscernibles: \(sim(x,x) = 0\)
3. Symmetry: \(sim(x,y) = sim(y,x)\)
4. Triangle-inequality: \(sim(x,z) \leq sim(x,y) + sim(y,z)\)

The (Damerau-)Levenshtein metrics have been assessed on these properties and only the Levenshtein edit logic satisfies all these conditions. The logic is extended in two ways in this research. First, the Dijkstra algorithm is applied to find a shortest path through the matrix. Second, the outcome of this shortest path is applied to measure temporal differences between sequences of activities. These extensions may pose a threat on the substantiation of the proposed metric to be a similarity metric. In section 6.1.1, some explanation is provided.
that supports conditions 1 and 2. Intuitively, the third criterion would receive support as an insert operation is the opposite of a delete. However, future work would mainly be focused on the criterion of triangle-inequality. This criterion allows to quickly reach conversion in a search with the concept of closed sets. It is achieved since triangle inequality allows to define the topology in a metric space. This criterion might not be satisfied due to the logic of the determination of the shortest path towards the final edit distance in the bottom right corner of the distance matrix. In this research, a shortest path is determined by assessing adjacent nodes on their costs in a clockwise order, starting in the direction down. Future work may consist of assessing the proposed metric on its validity as a similarity metric.

6.5.2 The proposed metric.

In the proposed metric parameters A-D may adjust to environments with different priorities with respect to average starting time and average duration of activities in a sequence. These properties might make the proposed metric suitable for application in many similarity assessment areas throughout a variety of (business) domains. However, now consider two sequences that are different from each other to the extent that the sum of delete, insert and substitute operations is equal to the total number of operations to obtain one sequence from the other. If parameter E is not equal to one, the similarity score of these sequences that are composed of completely different activities would not equal zero. Albeit the parameters are trained according to the dataset the metric is used in, parameter E will only be one if the other parameters are zero. This may imply that the proposed metric would not fully accommodate the lower end of similarity score calculation between activity sequence pairs. Conclusively, the choice of including parameter E in the proposed metric poses a threat for the low end of similarity matching when comparing fully distinct activity sequences. Future work may consist of re-evaluating which trainable parameters should, or should not be included into the proposed metric. It is suggested to investigate the accuracy of the following similarity score suggestedScore, which is the same as the proposed, but without the inclusion of parameter E. Applying this logic would yield a similarity score of 0 when the sum of the delete, insert and substitute edit operations equal the total number of edit operations, regardless of the training outcome.

\[
suggestedScore = 1 - \left( \frac{|M|}{|O|} \cdot (A \cdot \Delta ST_M + B \cdot \Delta DT_M) + \frac{|T|}{|O|} \cdot (C \cdot \Delta ST_T + D \cdot \Delta DT_T) + \frac{|D| + |I| + |S|}{|O|} \right)
\]

Further, the (Damerau-)Levenshtein logic is used to assess the differences from one activity sequence to another. It might be the case that if another logic would have been used, the findings would have been different. Future work may consist of assessing the effect of using different similarity measurement logics. The presented work has for instance a focus on shifting individual activities, whereas some application areas may benefit from identifying groups of activities. This may be achieved with for instance the n-grams logic as described by Barrón-Cedeño et al. (2010). This logic might result in higher accuracy values in some areas with recurring groups of activities, albeit the chance of an increased computational time.
The current model allows to process around 85 comparisons per second for lengths between eight and twelve activities on a i5-4310U Intel processor at 2.00GHz with 8GB RAM memory. Training on very large databases may thus be a time consuming effort. Future work could therefore consist of making the logic more efficient.

In this research, the edit operation costs of delete, insert and substitute are regarded the same in the construction of the distance matrix. Although the variables resulting from calculations based on these operations are parametrized in the proposed metric, they still determine the edge costs that influence the shortest path determination. Future work might consist of investigating whether changing the cost of the edit operations would yield more appropriate sets of edit operations.

6.5.3 Descriptive similarity assessment of truck routing schedules.

Although warning signs about inferring rules from descriptive data (McNally and Rindt, 2007) and low expected training accuracy with little side information available (Xing et al., 2003), the accuracy of the proposed metric has been slightly higher than random for the Damerau-Levenshtein based algorithm. The accuracy could still be seen as poor as the monkey based Levenshtein based test results outperformed the other assessed scenarios. This may for instance be due to the used data or the determination of the accuracy itself.

In a relatively early stage of this research, a chicken and egg problem arose with the selection of the data. The problem arose that a subset of the dataset of routing schedules should be created. Although a high number of routes was available, the amount of variation across these routes was high to such extent that the ratio of matched elements may have been on the low side. This may be a reason why the accuracy of the proposed metric is not very high, as it has a preference for matched elements. Future work may consist of assessing the metric in application areas with a more logical order of activities, or a higher level of matched operations. An example of such area is a production area, with a limited number of sequence orders and a clear input and output. In all, the accuracy of the proposed metric across the assessed routes has been low, which may have had to do with the ratio of matched elements. However, although the differences were small, the overall accuracy of the proposed metric was better than random. Besides, it has been shown that an increased match ratio does show some accuracy improvement for the Levenshtein based logic.

Another reason for the low accuracy value is the determination of this value itself. The method to decide on the accuracy value in this study might be too strict in two ways. First, imagine that if the proposed similarity ranking of a set of routes is one position off the expert input, say route one is at position two and route two is at position one, both route one and route two are only one position away from their expert judged counterparts. Therefore, the mis-ranking is counted for twice, resulting in accuracy value $1 - (10/12) = 83\%$. Second, the last few ranked routes may show a similar low resemblance with the target route. A difference in position there may not be of the same priority as for the first routes. However in the used measure, the accuracy is punished equally, regardless of where the deviation of the expert input is found. Therefore, future work may cover the assessment of the accuracy determination. The double-punishment may be solved by applying a proposed precedence-based accuracy measurement that has been developed as a by-product of this thesis. The elaborated procedure is displayed in Appendix C. The assessment of any ranking accuracy may further be extended with some priority rule where activity sequences on the higher judged ranking positions are valued more than on the lower ranking positions. Alternatively, the training could be approached differently than by ranking on similarity. For instance,
the activity sequences could be trained with classifying a rankee as high, medium or low similar. Another suggested approach is to ask experts to give an explicit similarity value. Conclusively, the choice of training may be dependent on the context that the metric is used in.

6.5.4 Relevance assessment.

The proposed metric has the potential to be used in search queries for (sections of) activity sequences in a business context. Further, a similarity based decision support tool for activity sequence improvements can potentially be beneficial in a business context. However, the conceptual descriptions are still far from mature. Below, some directions for future research are outlined.

- The proposed metric may be an efficient and fast search method for improvement suggestions when a database is constructed that defines all activity sequences in a specific domain. It may be further investigated whether using similarity as a starting point for improvement suggestions is indeed a time-efficient and effective approach.

- The proposed incremental solutions providing support tool may lead to local optima as similarity based search looks for much alike activity sequences. Future work could consist of changing the candidate selection such that high quality sequences that are not in the direct vicinity of the candidate solution are also evaluated.

- In a deterministic environment, the proposed mechanisms would function well. However, the mechanism does not account for variation across identical solutions. Also, the historical data-based approach may only work with large datasets of already performed and evaluated activity sequences. Future work may consist of elaborating on the boundary conditions for such mechanism to effectively function in a business environment.

- The suggestion to use genetic programming as a solution generator requires the search through a topological space. This topological space can only be achieved if all criteria of a similarity metric are met. Future work could therefore consist of first investigating the validity of the proposed metric as a similarity metric, and then to explore the possibilities of machine learning in solution generation.

- The potential of genetic programming in combination with the proposed SDSS has been described. However, in this research, it has not been investigated if activity sequences can be improved with genetic programming nor if feature selection is appropriate. Future work could consist of the applicability of genetic programming in this domain.

- In literature, there is a lot of debate around automation in planning and scheduling. Higgins (1999) opts for a hybrid intelligent human-computer scheduling paradigm, in which human and machine intelligence are combined. Many scholars question whether all factors influencing the construction, both objective and subjective, of a schedule can be automated. Future work could consist of investigating what kind of activity sequences are suitable for automated sequence invention.
Bibliography


50


Appendix A

Instruction Text Questionnaire

Dear [participant],

In the following series of questions, you will review 15 sets of transportation routing schedules. The coloured bars represent the stay time at a specific location, from arrival to leaving. The different colors represent anonymized real unloading points. The white spaces between the coloured bars indicate the travel time from one location to another.

You have been invited for these comparison exercises because of your knowledge and experience in constructing and evaluating schedules. The center of gravity of this research lies in training the similarity model according to your expert opinion as closely as possible.

What is expected from you?
The routing schedules should be regarded as "unconstrained" in the comparison. This means that if the staytime is longer, the route was expected to cost more time, due to a higher unloading quantity or planner experience. If the space between two location is longer, you can assume that the travel time/distance is bigger. I would kindly like to invite you to rank the structure of the planning structure to the target schedule in each following schedule to your best intuition. I invoke your experience in scheduling to identify when a schedule is more similar to the target than another; be it for instance:

- Similar structure/patterns
- Comparable locations
- Starting time

How?
Drag and drop the responses or use the number drop down menus to rank the routing schedules.

Where does the data come from?
The data comes from a large truck operator in the Netherlands. The data has been anonymized and generalized.

Why is this research relevant?
Apart from the scientific relevance and novelty of being able to compare series of activities to one another, this research may serve as an efficient approach to compare and improve schedules as a decision support tool. The data we are currently testing on, is the expected staytime in the locations. If we know how similar one schedule is to another by mimicking
expert judgement, we can estimate the expected performance, quickly adapt to changes, propose better solutions etc.
Appendix B

Participants Description

In this chapter, the experts that participated in the research are described. Most expert roles have been taken performed for clients of Accenture. Details of client projects are confidential and therefore their names are not noted explicitly.

Participant 1

Current position: Consultant Supply Chain & Operations, Accenture Strategy, Warsaw
Specialization: Demand, Supply, Production & Inventory Planning as well as Supply Chain Software Implementation
Highest education: Master Global Production Engineering and Management

Table B.1: Experience Summary Participant 1

<table>
<thead>
<tr>
<th>Company</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global FMCG</td>
<td>Leader</td>
</tr>
<tr>
<td>Continental</td>
<td>Explore Tires Logistics Trainee Program</td>
</tr>
</tbody>
</table>

Participant 2

Current position: Manager Supply Chain Management, Accenture Management Consulting UK
Specialization: Demand and Supply Planning expert within the Fast Moving Consumer Goods Industry
Highest education: Bachelor Business Management

Table B.2: Experience Summary Participant 2

<table>
<thead>
<tr>
<th>Company</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procter &amp; Gamble</td>
<td>Sales &amp; Operations Planning and Demand Planning Capability Leader</td>
</tr>
<tr>
<td>Procter &amp; Gamble</td>
<td>Market Planning Team Leader</td>
</tr>
</tbody>
</table>
Participant 3
Current position: Senior Manager Supply Chain & Operations, Accenture Strategy Chicago
Specialization: Integrated Supply Chain Planning Capability
Highest education: Master Supply Chain Management Technology

Table B.3: Experience Summary Participant 3

<table>
<thead>
<tr>
<th>Company</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accenture</td>
<td>Subject Matter Expert in Sales Operation Planning, Demand Planning and Supply Planning</td>
</tr>
<tr>
<td>Beauty Retailer</td>
<td>Integrated Planning Business Transformation Lead</td>
</tr>
<tr>
<td>Large Canadian Retailer</td>
<td>Supply Chain Business Lead Architect</td>
</tr>
<tr>
<td>Regional Retailer</td>
<td>Supply Chain Planning Lead</td>
</tr>
</tbody>
</table>

Participant 4
Current position: Senior Manager Functional & Industry Analytics, Accenture Digital Barcelona
Specialization: Forecasting
Highest education: Master Mathematics and Financial Mathematics

Table B.4: Experience Summary Participant 4

<table>
<thead>
<tr>
<th>Company</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large CPG Company</td>
<td>Analytics Team Lead for Initial Pilot and Subsequent Forecasting Managed Service</td>
</tr>
<tr>
<td>Large Consumer Electronics Manufacturer</td>
<td>Service Manager and Chief Modeler for Demand Forecast Managed Service</td>
</tr>
</tbody>
</table>

Participant 5
Current position: Consultant Supply Chain & Operations, Accenture Strategy Warsaw
Specialization: Supply Chain Planning and Fulfillment
Highest education: Master Management & Organization of Production

Table B.5: Experience Summary Participant 5

<table>
<thead>
<tr>
<th>Company</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Food and Beverage Company</td>
<td>Supply Chain Planning Operating model</td>
</tr>
<tr>
<td>Global AlcoBev Company</td>
<td>Supply Planning &amp; Forecasting</td>
</tr>
<tr>
<td>Delitissue (Softidel Group)</td>
<td>Planning &amp; Procurement Manager.</td>
</tr>
</tbody>
</table>
Participant 6

Current position: Manager at Consumer Goods & Retail Practice of Accenture Consulting
Specialization: Supply Chain Planning (Demand and Supply Planning)
Highest education: Master International Relations, International Trade

Table B.6: Experience Summary Participant 6

<table>
<thead>
<tr>
<th>Company</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global AlcoBev company</td>
<td>Strategy Consulting in Supply Planning</td>
</tr>
<tr>
<td>Lead Subject Matter Expert</td>
<td>Supply Planning</td>
</tr>
<tr>
<td>Hewlett Packard</td>
<td>Project Manager SAP Roll-out, Internal Consultant</td>
</tr>
</tbody>
</table>

Participant 7

Current position: Consultant Supply Chain & Operations, Accenture Strategy Warsaw
Specialization: Supply Chain Management GEEK with Focus on Supply Planning/Optimization, Warehouse Management and With a Good Friendship With Lean six Sigma (Green Belt)
Highest education: Master in Engineering - Logistics Audit

Table B.7: Experience Summary Participant 6

<table>
<thead>
<tr>
<th>Company</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accenture</td>
<td>Business Subject Matter Expert for two top Tier SCM Solutions - JDA and Kinaxis</td>
</tr>
<tr>
<td>Mondelēz Europe Services GmbH</td>
<td>Supply Planning Manager</td>
</tr>
<tr>
<td>Avon Cosmetics Poland</td>
<td>Supply Chain Analyst</td>
</tr>
</tbody>
</table>
Appendix C

Proposed Precedence-Based Ranking Accuracy Determination Procedure

In this section, a ranking accuracy determination procedure is proposed. It is a by-product of this thesis but may be useful for further analysis. Wall (1996) has functioned as a source of inspiration in the making of this procedure one one of the products of his dissertation has been a precedence -ased task order comparison. Future work could exist of working out the mechanism and applying it in training the proposed metric or leveraging it in other areas. The method should be further examined but some prototyping is displayed in this appendix. Summary: The presented ranking accuracy determination allows to compare two rankings based on precedence relationships among the ranked elements. The main difference with the accuracy measure used in this thesis is that it is a less strict accuracy measurement in the sense that it costs only one step to reverse a pair of switched ranking items.

Maximum Ranking Difference Steps

In Table C.1, the maximum steps of reaching sequence 5-4-3-2-1 from 1-2-3-4-5 are outlined where a step is defined as moving one ranked item one position at a time. It can be concluded that following this logic, the maximum distance MaxDist for five ranking positions is 10 steps.
Table C.1: Maximum Steps Proposed Ranking Accuracy Determination With Five Ranking Positions

<table>
<thead>
<tr>
<th>Step</th>
<th>Pos 1</th>
<th>Pos 2</th>
<th>Pos 3</th>
<th>Pos 4</th>
<th>Pos 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Displaying a Ranking in a Precedence-Based Matrix

In a matrix form, a sequence can be displayed in a precedence-based matrix. Consider the first row in sequence 1-2-3-4-5 as displayed in Table C.2. After item named 1, 2-5 follow so those cells receive value 1. After value 2 follow 3-5, so row 2 column 3-5 are also filled with value one.

Table C.2: Precedence-Based Representation of Ranking A:1-2-3-4-5

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Comparison of Two Ranking Accuracy Measures

In Table C.3, four example, precedence-based ranking representations are displayed. In this thesis, it is proposed that the sum of the product of two of these matrices $M_i$, divided by the maximum distance.
Table C.3: Four Example Precedence-Based Ranking Representations

<table>
<thead>
<tr>
<th></th>
<th>A: 1-2-3-4-5</th>
<th>C: 1-3-2-5-4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>1</td>
<td>1 1 1 1 1</td>
<td>1 1 1 1 1</td>
</tr>
<tr>
<td>2</td>
<td>1 1 1 2</td>
<td>1 1</td>
</tr>
<tr>
<td>3</td>
<td>1 1 3</td>
<td>1 1 1</td>
</tr>
<tr>
<td>4</td>
<td>1 4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5 1</td>
<td>1</td>
</tr>
<tr>
<td>B:</td>
<td>5-4-3-2-1</td>
<td>D: 3-5-2-4-1</td>
</tr>
<tr>
<td></td>
<td>1 2 3 4 5</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1 2 1 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1 1 1 1 1</td>
<td>3 1 1 1 1</td>
</tr>
<tr>
<td>4</td>
<td>1 1 1 4</td>
<td>4 1</td>
</tr>
<tr>
<td>5</td>
<td>1 1 1 1 5</td>
<td>1 1 1 1 1</td>
</tr>
</tbody>
</table>

The accuracy measure of two ranking sequences is thus defined as:

\[ \text{RankAcc} = \frac{\sum M_i \times M_j}{\text{MaxDist}} \]

In Table C.4, this RankAcc is calculated for all pairs of rankings associated with sequence A.

Table C.4: Example Proposed RankAcc With \( M_A \)

<table>
<thead>
<tr>
<th></th>
<th>RankAcc with ( M_A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_A )</td>
<td>100%</td>
</tr>
<tr>
<td>( M_B )</td>
<td>0%</td>
</tr>
<tr>
<td>( M_C )</td>
<td>80%</td>
</tr>
<tr>
<td>( M_D )</td>
<td>30%</td>
</tr>
</tbody>
</table>

In Table C.5, the absolute position distance and accuracy between sequence A and sequences B-D is displayed. The accuracy \( \text{AbsAcc} \) is obtained by dividing the absolute position distance by the maximum distance, which is has been shown in Table 4.1 to be 12 for a sequence of five ranking positions.

Table C.5: Absolute Position Distance

<table>
<thead>
<tr>
<th>Tested sequence</th>
<th>Absolute position distances from sequence A: 1-2-3-4-5</th>
<th>AbsAcc with seq. A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A: 1-2-3-4-5</td>
<td>0+0+0+0+0=0</td>
<td>100%</td>
</tr>
<tr>
<td>B: 5-4-3-2-1</td>
<td>4+2+0+2+4=12</td>
<td>0 %</td>
</tr>
<tr>
<td>C: 1-3-2-5-4</td>
<td>0+1+1+1+1=4</td>
<td>67%</td>
</tr>
<tr>
<td>D: 3-5-2-4-1</td>
<td>2+3+1+0+4=10</td>
<td>17%</td>
</tr>
</tbody>
</table>

Clearly, both accuracy measures yield two different accuracy scores but can both be defined as an objective measure to judge the accuracy of two different rankings. Naturally,
the choice of measurement has an influence on the outcome of rank-based training.