Joh, Chang-Hyeon; Arentze, Theo A.; Timmermans, Harry J. P.

Conference Paper

Multidimensional Sequence Alignment Methods for Activity Pattern Analysis: A comparison of dynamic programming and genetic algorithms

39th Congress of the European Regional Science Association: "Regional Cohesion and Competitiveness in 21st Century Europe", August 23 - 27, 1999, Dublin, Ireland

Provided in Cooperation with:
European Regional Science Association (ERSA)

Suggested Citation: Joh, Chang-Hyeon; Arentze, Theo A.; Timmermans, Harry J. P. (1999) : Multidimensional Sequence Alignment Methods for Activity Pattern Analysis: A comparison of dynamic programming and genetic algorithms, 39th Congress of the European Regional Science Association: "Regional Cohesion and Competitiveness in 21st Century Europe", August 23 - 27, 1999, Dublin, Ireland, European Regional Science Association (ERSA), Louvain-la-Neuve

This Version is available at:
https://hdl.handle.net/10419/114358

Multidimensional Sequence Alignment Methods for Activity Pattern Analysis:
A comparison of dynamic programming and genetic algorithms

Chang-Hyeon Joh, Theo A. Arentze and Harry J.P. Timmermans[1]

Urban Planning Group
Eindhoven University of Technology
P.O. Box 513, Mail Station 20, 5600 MB, Eindhoven, The Netherlands, C.H.Joh@bwk.tue.nl

Abstract---Quantitative comparisons of space-time activity patterns are a critical element in several streams of research in regional science. Traditionally, Euclidean distance and the measures developed in botanical taxonomy have been widely used to measure the similarity between activity patterns that involve several attribute dimensions such as location, transport mode, accompanying persons, etc. Some other techniques, such as pattern recognition in signal processing theory, have also been introduced for this purpose. These measures however lack the ability to capture the information of the overall sequence of activity patterns of multiple attributes. Recently, the Sequence Alignment Methods (SAMs), developed in molecular biology that are concerned with the distances between DNA strings, have been introduced in time use research. The SAMs captures the similarity of activity patterns based on a single attribute only. Unfortunately, the extension of the uni-dimensional SAM to a multidimensional method induces the problem of combinatorial explosion. To solve this problem, this paper introduces effective heuristic methods for the comparison of multidimensional activity patterns. First, the combinatorial nature of the algorithm is discussed. The paper then develops alternative SAMs based on dynamic programming and genetic algorithms, respectively. These two SAMs are compared using empirical activity pattern data. The paper ends by discussing avenues of future research.

1. Introduction

Quantitative comparisons of space-time activity patterns are a critical element in several streams of research in regional science, including activity pattern classification and simulation-based travel demand forecasting. In classification studies, for example, a matrix of pairwise distances between observed activity patterns is used as a similarity measure, providing the input for a subsequent cluster analysis. The resulting classification is then typically correlated with a set of spatial and/or socio-economic variables of interest. In activity-pattern simulation modeling, the distance between observed and predicted activity patterns is used to assess the goodness-of-fit of the model. These activity-based approaches are rapidly gaining increasing attention, particularly in travel demand-forecasting research (e.g., Ettema and Timmermans, 1997). Because these approaches focus on the relationships between particular aspects of activities and specific types of trips, the similarity between activity patterns is typically measured by multiple variables, describing the pluriform nature of activity patterns.

Regardless of the objectives of the study, the measurement of similarity of activity patterns involves operational decisions about *what* to compare and *how* to quantify the similarity. Reflecting the notion that choices of (sets of) destinations, travel modes, departure times and routes are likely to be made simultaneously (Gärling *et al*., 1997), a similarity measure that can compare multiple elements at a time would be most desirable. Moreover, because activity

[1] Chang-Hyeon Joh is Ph.D. candidate at the Eindhoven University of Technology. Theo Arentze is postdoctorate at the Eindhoven University of Technology. Harry Timmermans is chaired professor of Urban Planning Group at the Eindhoven University of Technology.

Paper prepared for presentation at the 39th ERSA Conference in Dublin, Ireland in August 1999.

patterns involve by definition a sequence of activities, the similarity or goodness-of-fit measure should allow one to measure sequential differences.

Previous measures can be classified into three different sets of measures. The first set considers mainly the difference in element composition between activity patterns and compares pairs of corresponding elements in two activity patterns. An example is Gower's similarity index (Burnett and Hanson, 1982; Pas, 1983), originally developed in botanical taxonomy (Gower, 1971). The elements of corresponding positions of two activity patterns are compared for each choice dimension or attribute. The sum of such differences across choice dimensions is taken as a measure of similarity between the two activity patterns. This position-based Euclidean distance measure does not capture the sequential information embedded in each attribute sequence nor recognizes, if any, the bundles of elements that can be compared concurrently.

The second set of measures is concerned with differences in both element composition and sequential order between activity patterns, but compares each element separately. An example is the feature extraction method based on the Walsh/Hadamard transformation (Recker *et al.*, 1985; Golob and Recker, 1989), originally developed in signal processing theory (Young and Calvert, 1974). Each activity pattern is encoded by a pattern matrix consisting of column vectors of attribute elements that respectively denote the temporal variation of the distance from home and binary indices of different activity types and transport modes. Each column vector is then transformed into a column of coefficients, each characterizing the element's sequence in terms of the degree of the correspondence against the sequences of various binary Walsh functions. Finally, the column vectors of Walsh/Hadamard-transformed attribute elements are cluster-analyzed to classify the activity patterns. This measure takes the sum of the degree of difference in individual coefficients between attribute elements as a measure of similarity of overall distance between activity patterns. Because the results of these comparisons are summed independently, this approach excludes the possibility of collective comparison of multiple elements across different attributes between activity patterns.

Unlike the previous measures, the measure to be developed in this paper, is concerned with differences in both element composition and the sequential order of elements between activity patterns, and compares bundles of elements across attribute dimensions simultaneously. This approach is inspired by the introduction of a sequence alignment method (Wilson, 1998) in time use research. This distance measure originally developed in molecular biology (Kruskal, 1983) uses the smallest number of changes (mutations) required to equalize two biological sequences of information such as DNA, RNA or proteins as being a measure of (dis)similarity between the sequences. Unfortunately, however, the existing methods can handle uni-dimensional sequences only. A simple sum of the results of the conventional uni-dimensional sequence alignment methods does not capture the sequential information of set of elements across different attributes. In a previous paper, we therefore developed a multidimensional extension of the sequence alignment method to capture the sequential information as well as the accompanying relationships in multidimensional activity patterns (Joh, *et al*., 1997).

Although this measure is theoretically appealing, it has the disadvantage of very high computing times, which is the result of the combinatorial nature of the multidimensional comparison (Arentze, *et al*., 1998). The current paper therefore explores the possibility of developing effective algorithms to compute a similarity measure in real time by comparing the performances of multidimensional comparison methods based on different heuristics. These algorithms were developed as part of the ALBATROSS project[2] that seeks to develop an activity-based model of transport demand in terms of a rule-based simulation model of activity scheduling behavior.

The remainder of the paper is organized into four sections. Section 2 describes the combinatorial nature of the multidimensional activity pattern-comparison problem. Having

[2] ALBATROSS stands for A Learning-Based Transportation Oriented Simulation System.

explained the nature of the problem, the paper then discusses two alternative heuristic approaches, one based on dynamic programming and the other on genetic algorithms. Section 4 then illustrates and compares the performances of these two approaches, using activity pattern data collected in The Netherlands. The paper ends with a conclusion and discussions.

2. The Problem

Before discussing the algorithms we will briefly introduce the multidimensional pattern comparison problem.

2.1 Multidimensional pattern comparison problem

Let us assume that the comparison of multidimensional activity patterns involves K qualitative attributes. These attributes may include activity type, location, transport mode, accompanying person, etc. Thus, each *activity pattern* consists of K *attribute sequences*, and each attribute sequence consists of a set of *elements*. To limit the discussion, we do not concern ourselves in this paper with quantitative attributes, such as activity duration.

Each attribute constitutes a sequence, whose elements can be represented by a set of characters. Two activity patterns to be compared can then be represented by a pair of K-dimensional source and target patterns.

source pattern $\boldsymbol{s} = \boldsymbol{s}[\boldsymbol{s}_1 \ldots \boldsymbol{s}_k \ldots \boldsymbol{s}_K]'$

with

$\boldsymbol{s}_k. = \boldsymbol{s}_k.[s_{k0} \ldots s_{ki} \ldots s_{km}]$

and

target pattern $\boldsymbol{g} = \boldsymbol{g}[\boldsymbol{g}_1 \ldots \boldsymbol{g}_k \ldots \boldsymbol{g}_K]'$

with

$\boldsymbol{g}_k. = \boldsymbol{g}_k.[g_{k0} \ldots g_{kj} \ldots g_{kn}]$

where,

$\boldsymbol{s}_k.$ and $\boldsymbol{g}_k.$ are the vectors of the k-th attribute sequences of the source and target patterns, respectively;

m and n are respectively the number of elements in the source and target activity patterns;

s_{ki} and g_{kj} are the i-th and j-th element of the k-th attribute sequence of the source and target patterns, respectively ($i = 0, \ldots, m$; $j = 0, \ldots, n$; $k = 1, \ldots, K$); $s_{ki} \in A_k$, and $g_{kj} \in A_k$.

Consequently, column vectors $\boldsymbol{s}._i = \boldsymbol{s}._i[s_{1i} \ldots s_{ki} \ldots s_{Ki}]'$ and $\boldsymbol{g}._j = \boldsymbol{g}._j[g_{1j} \ldots g_{kj} \ldots g_{Kj}]'$ represent activity type, location, transport mode, etc. of the i-th and j-th activity, respectively. In particular, when the number of concerned attributes is given 1 (that is, $K = 1$), the comparison problem becomes uni-dimensional, and there exist solutions, one of which known as *Levenshtein distance* that is defined as the smallest number of changes made on the elements to equalize two sequences. A set of equations defining the 'weighted' Levenshtein distance is:

$$d(\boldsymbol{s}_k, \boldsymbol{g}_k) = d(\boldsymbol{s}_{km}, \boldsymbol{g}_{kn})$$
$$d(\boldsymbol{s}_{k0}, \boldsymbol{g}_{k0}) = 0$$
$$d(\boldsymbol{s}_{ki}, \boldsymbol{g}_{k0}) = d(\boldsymbol{s}_{ki\text{-}1}, \boldsymbol{g}_{k0}) + w_d(s_{ki}, f) \qquad i \geq 1$$

$$d(\boldsymbol{s}_{k0}, \boldsymbol{g}_{kj}) = d(\boldsymbol{s}_{k0}, \boldsymbol{g}_{kj\text{-}1}) + w_i(f, g_{kj}) \qquad j \geq 1$$

$$d(\boldsymbol{s}_{ki}, \boldsymbol{g}_{kj}) = \min[d(\boldsymbol{s}_{ki\text{-}1}, \boldsymbol{g}_{kj}) + w_d(s_{ki}, f), d(\boldsymbol{s}_{ki}, \boldsymbol{g}_{kj\text{-}1}) + w_i(f, g_{kj}), d(\boldsymbol{s}_{ki\text{-}1}, \boldsymbol{g}_{kj\text{-}1}) + w(s_{ki}, g_{kj})] \; i,j \geq 1$$

with

$$w(s_{ki}, g_{kj}) = \begin{cases} w_e(s_{ki}, g_{kj}) = 0 & \text{if} \quad s_{ki} = g_{kj} \\ w_s(s_{ki}, g_{kj}) = d[w_d(s_{ki}, f) + w_i(f, g_{kj})] & \text{otherwise} \end{cases}$$

where,

$\boldsymbol{s}_{ki}$ and $\boldsymbol{g}_{kj}$ are initial parts of $\boldsymbol{s}_k$ and $\boldsymbol{g}_k$ ($\boldsymbol{s}_{ki}$=$\boldsymbol{s}_{ki}[s_{k0} \ldots s_{ki}]$, and $\boldsymbol{g}_{kj}$=$\boldsymbol{g}_{kj}[g_{k0} \ldots g_{kj}]$, where $0 \leq i \leq m$, and $0 \leq j \leq n$);

$d(\boldsymbol{s}_{ki},\boldsymbol{g}_{kj})$ is the cost of equalization of $\boldsymbol{s}_{ki}$ with $\boldsymbol{g}_{kj}$, cumulated from the equalization of $\boldsymbol{s}_{k0}$ with $\boldsymbol{g}_{k0}$;

$d(\boldsymbol{s}_k,\boldsymbol{g}_k)$ is the cost of equalization of $\boldsymbol{s}_k$ (= $\boldsymbol{s}_{km}$) with $\boldsymbol{g}_k$ (= $\boldsymbol{g}_{kn}$); $w_d(s_{ki}, f)$ is the cost for deleting the i-th element of s_k;

$w_i(f,g_{kj})$ is the cost for inserting the j-th element of g_k; $w(s_{ki},g_{kj})$ is the cost for replacing s_{ki} with g_{kj} (Subscripts e and s in $w_e(s_{ki},g_{kj})$ and $w_s(s_{ki},g_{kj})$ indicate identity (*e*quality) and substitution, respectively.);

d is a substitution coefficient.

$d(\boldsymbol{s}_{ki},\boldsymbol{g}_{kj})$ is therefore determined by the minimum among the cumulative costs for the deletion of s_{ki} from $\boldsymbol{s}_{ki}$, the insertion of g_{kj} from $\boldsymbol{g}_{kj}$ and the replacement (substitution or identity) of s_{ki} in $\boldsymbol{s}_{ki}$ with g_{kj} in $\boldsymbol{g}_{kj}$. An optimum alignment consists of an ordered set of operations (deletions, insertions, substitutions and identities) applied to equalization of the initial parts of $\boldsymbol{s}_k$ and $\boldsymbol{g}_k$, from $\boldsymbol{s}_{ki}$-$\boldsymbol{g}_{kj}$ pair to $\boldsymbol{s}_{km}$-$\boldsymbol{g}_{kn}$ pair. Note that there are often many ways to optimally align a pair of sequence, as the cumulative costs to determine $d(\boldsymbol{s}_{ki},\boldsymbol{g}_{kj})$ are often the same between deletion, insertion and replacement. (A detailed explanation of the uni-dimensional sequence alignment methods can be found in e.g. Kruskal (1983) and Arentze *et al*. (1998).)

A multidimensional problem of interest in this paper is, however, concerned with two or more attribute sequences (that is, $K \geq 2$). In such cases, two sequences $\boldsymbol{s}_k$. and $\boldsymbol{g}_k$. can be compared if and only if $k = k'$ (the same attribute). Furthermore, the number of elements may differ *between* patterns (that is, $m = n$ or $m \neq n$), but is the same *within* a pattern ($km = k'm$, and $kn = k'n \quad \forall\ s_{kn}, g_{kn}$). The problem of multidimensional activity-pattern comparison then is to find the minimum amount of effort required to change $\boldsymbol{s} = \boldsymbol{s}[\boldsymbol{s}_1 \ldots \boldsymbol{s}_K]'$ into $\boldsymbol{g} = \boldsymbol{g}[\boldsymbol{g}_1 \ldots \boldsymbol{g}_K]'$ in the multidimensional space.

When determining the similarity of multidimensional activity patterns, different strategies may be adopted, dependent on whether attributes are considered separately or as a set. The MultiDimensional Sequence Alignment Method (MDSAM) proposed in this paper intends to incorporate both strategies. The choice of a particular strategy varies with the alignment situation during the comparison. To clarify our assumptions underlying the intended properties of the MDSAM, consider two special cases. In the first special case, the attributes are perfectly independent in the sense that the operations are applied independently to attributes. In this case, the MDSAM can be defined as the sum of Uni-Dimensional Sequence Alignment Methods (UDSAMs). In equation:

$$d(\boldsymbol{s},\boldsymbol{g}) = \sum_{k=1}^{K} b_k d(\boldsymbol{s}_k, \boldsymbol{g}_k) \qquad (1)$$

where,

$d(\boldsymbol{s},\boldsymbol{g})$ is a measure of the distance between $\boldsymbol{s}$ and $\boldsymbol{g}$;

$\boldsymbol{s}_k$ and $\boldsymbol{g}_k$ are the k-th attribute sequence of $\boldsymbol{s}$ and $\boldsymbol{g}$, respectively;
$d(\boldsymbol{s}_k,\boldsymbol{g}_k)$ is the weighted Levenshtein distance between $\boldsymbol{s}_k$ and $\boldsymbol{g}_k$;
b_k is the weight of the k-th attribute sequence.

The weights, b_k, may be different from attribute to attribute in the sense that certain activity attributes, such as activity type, may be considered more important than others. Nevertheless, this independent alignment case assumes that decisions regarding a particular attribute are made independently from all other attributes.

In the second special case, all attributes are perfectly dependent in the sense that each operation can be applied jointly to all K elements. This would be the case if, for example, each activity type that is common to the compared activity patterns is accompanied by exactly the same set of elements across all other dimensions between activity patterns compared. In this case, the same operation can be applied jointly to the set of K elements. To deal with these perfect contingent relationships between elements of different attributes, we introduce the concept of 'a segment,' defined as a set of elements of different attributes that can be aligned simultaneously.

Aligning several elements of different attributes at the same time implies that the operations applied to each element in a segment are integrated across attributes. The problem then is how to determine the aggregate weighting value of the segment. Unless the weights of K attributes are all the same, the alignment costs depend on the aggregation method used. We argue that the maximum weight across attributes of which the elements are aligned together may be the most appropriate alternative for our purpose. This reflects the notion that people's aggregate decision in activity scheduling may be dominated by the most important attribute. The MDSAM for the perfect dependency case can then be measured as the sum of segment-based operation weights:

$$d(\boldsymbol{s},\boldsymbol{g}) = \sum_{k=1}^{K} b^{\max} \frac{d(\boldsymbol{s}_k,\boldsymbol{g}_k)}{K} \tag{2}$$

where $b^{\max}$ is the maximum weight across K attributes. (Note that every segment includes K elements, one from each attribute, because each common activity type shares all other K–1 attribute elements between $\boldsymbol{s}$ and $\boldsymbol{g}$.)

These special cases are, however, extremes. In reality, attributes may well be partly dependent, which means that a segment may have less than K elements and represents a variety of accompanying relationships between attributes. In general, the MDSAM introduced in this paper will allow reductions in alignment costs by segment-based operations of accompanying attributes. The stronger the interdependency between attributes, the higher the reduction. For example, let O be a set of operations that is applied simultaneously to align attributes $k = k_1$ and $k = k_2$ with costs $d(\boldsymbol{s}_k,\boldsymbol{g}_k)$. Then, the costs of segment alignments would be $max(b_{k1},b_{k2})\times d(\boldsymbol{s}_k,\boldsymbol{g}_k)$. In the independent case, on the other hand, the costs would be $(b_{k1}+b_{k2})\times d(\boldsymbol{s}_k,\boldsymbol{g}_k)$. The cost reduction through the segment alignments is therefore $(b_{k1}+b_{k2})\times d(\boldsymbol{s}_k,\boldsymbol{g}_k) - max(b_{k1},b_{k2})\times d(\boldsymbol{s}_k,\boldsymbol{g}_k) = min(b_{k1},b_{k2})\times d(\boldsymbol{s}_k,\boldsymbol{g}_k)$. The similarity measure based on this MDSAM is then defined as the alignment costs that minimize the sum of uni-dimensional alignment costs, reduced by the costs saved because of the relationships between accompanying attributes.

In general, there will be many ways to align uni-dimensional attribute sequences. Consequently, the combination of uni-dimensional operations across attributes will result in a number of multidimensional operation sets that define which operations have been applied to equalize the multidimensional activity patterns. The similarity measure can therefore be defined as:

$$C^* = \min [C^1, \ldots, C^u, \ldots, C^{TU}] \tag{3}$$

with

$$C^u = \sum_{p \in O_u} c_p \tag{4}$$

and

$$c_p = \begin{cases} w_d \, b_{\boldsymbol{k}}^{\max} & \text{if } \; p = \mathtt{d}(i,\boldsymbol{k}) \\ w_i \, b_{\boldsymbol{k}}^{\max} & \text{if } \; p = \mathtt{i}(j,\boldsymbol{k}) \end{cases} \tag{5}$$

and

$$O_u = \{p | p = \mathtt{d}(i,\boldsymbol{k}) \vee \mathtt{i}(j,\boldsymbol{k})\} \tag{6}$$

where,

O_u is the u-th multidimensional operation set;
C^u is the alignment costs based on O_u;
TU is the number of all possible multidimensional operation sets;
w_d and w_i are the weighting values of deletion and insertion operations, respectively;
$\mathtt{d}(i,\boldsymbol{k})$ and $\mathtt{i}(j,\boldsymbol{k})$ are respectively the deletion and insertion operations applied to the i-th and j-th elements of the set of attribute dimensions included in the set $\boldsymbol{k}$;
$b_{\boldsymbol{k}}^{\max}$ is the maximum weight across attributes in the set $\boldsymbol{k}$.

We assume that $w_d = w_i$, and that the costs for substituting a source element with a target one is the same as the sum of the costs for deleting the source element and inserting the target element. Consequently, a substitution operation [$\mathtt{s}(i,j,\boldsymbol{k})$] is not included in equations (5) and (6), as it is always decomposable into a deletion and insertion. That is, $\mathtt{s}(i,j,\boldsymbol{k}) = \mathtt{d}(i,\boldsymbol{k}) + \mathtt{i}(i,\boldsymbol{k})$.

2.2 Combinatorial nature of the problem

Each set of elements sharing the same operation composes a segment in the activity pattern and is aligned as if it is one element. The MDSAM problem is then to find the set of segments inducing the smallest sum of operation weights. There is, however, no existing method that aligns all attributes at the same time and finds the segment sets that guarantee optimality. For example, one may try a multidimensional distance measure that first aligns each attribute and then integrates the resulting uni-dimensional operation sets across attribute dimensions. Unfortunately, this alignment procedure does not always provide the optimum solution. Consider for example the following comparison of two sequences of lengths m=4 and n=5, respectively.

source pattern	1 4 7 8	(attribute 1)
	1 4 2 3	(attribute 2)
target pattern	1 2 3 4 5	(attribute 1)
	1 2 3 4 5	(attribute 2)

Assuming that $b_1 = b_2 = 1$, and $w_d = w_i = 1$, the uni-dimensional optimum alignments can be represented in terms of comparison tables (Fig. 1). The uni-dimensional operation sets are {d4, d5, i3, i4, i6} and {d3, i5, i6} for attribute 1 and attribute 2 respectively, where, for example, d5 and i3 denote the deletion of the fifth source element and insertion of the third target element, respectively. The corresponding multidimensional operation set is O = {d3(2), d4(1), d5(1), i3(1), i4(1), i5(2), i6(1,2)}, which results in a multidimensional alignment cost of 7, using equations (4) and (5). However, by aligning the second attribute sequences

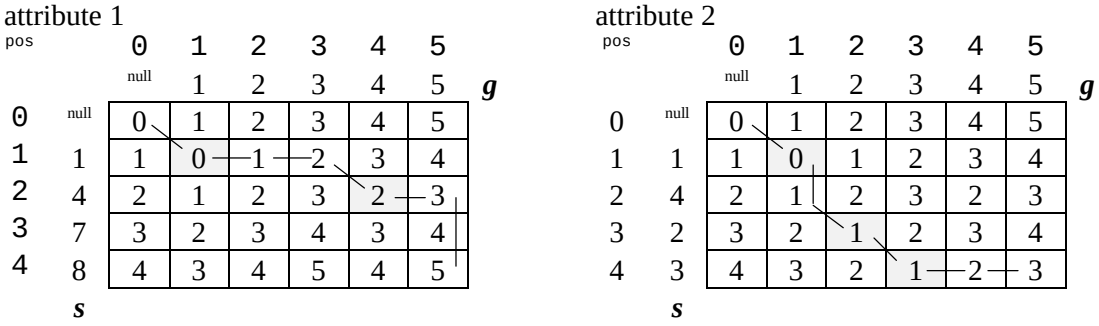

Fig. 1: Comparison tables of two attribute dimensions. The vertical, horizontal and diagonal moves represent the deletion, insertion and identity operations, respectively, and the ordered set of moves from cell (0,0) to (4,5) constitutes a trajectory of the alignments in each comparison table.

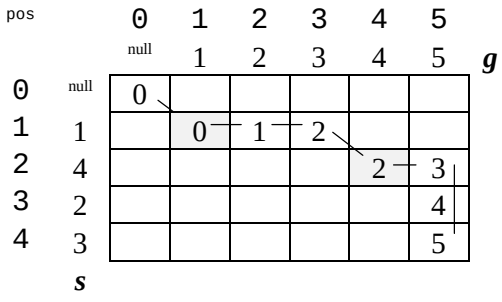

Fig. 2: A non-optimum trajectory of attribute 2.

differently as in Fig. 2, the uni-dimensional operation set for attribute 2 becomes {d4, d5, i3, i4, i6}, and we obtain the multidimensional operation set as O = {d4(1,2), d5(1,2), i3(1,2), i4(1,2), i6(1,2)}. The multidimensional alignment costs are therefore equal to 5. The alignment of attribute 2 in the second case is not optimal, and hence, the uni-dimensional alignment costs are bigger than the optimum. Thus, the non-optimal uni-dimensional alignment induces a smaller multidimensional alignment cost that is optimal.

In general, the accompanying relationships between attributes may be quite variable, leading to different MDSAM results that the integration of uni-dimensional optimum alignments cannot perfectly project. To date, the only known way to guarantee the optimality of the multidimensional comparison result is to try all possible alignments of each attribute, generate all possible combinations of the alignments across attributes and then find the minimum costs. Unfortunately, the enumeration of all possible alignments of even a single attribute already requires an enormous amount of computing time. In this paper, we therefore suggest employing heuristic approaches. In particular, two such approaches or algorithms will be developed in the following section.

3. Alternative Methods

Enumerating all possible solutions is, while guaranteeing the optimality of the final solution, not a realistic approach in terms of computing time. Given the nature of the problem, the alternative approach must seek acceptable results within acceptable computing times by reducing the solution search space. To accomplish this, two alternative heuristic algorithms are introduced in this section. One is a genetic algorithm, and the other is based on a dynamic programming technique used for uni-dimensional sequence alignments.

3.1 A heuristic method based on genetic algorithms

Genetic Algorithms (GA) consitute a heuristic approach, inspired by Darwinian evolution theory. Its *solution representation* and search mechanism is modeled in analogue to evolutionary processes of biological species. In this approach, a possible solution is represented as an *organism* consisting of one or more *chromosomes*. A chromosome is a string of *genes* containing genetic information. GA does not search the entire space of a possibly infinite number of solutions, but considers only a *population*: a pre-defined, tractable number of organisms. It starts with initializing the population (called the 0-th *generation*) by using a random generator and evaluates all organisms of the population according to some mathematical function, resulting in a fitness value of each organism. A certain number of organisms are then probabilistically selected from the population, based on their fitness values. The higher the fitness of an organism, the greater the probability of being selected. GA then generates a new population by applying genetic operators such as reproduction, crossover and mutation to the selected organisms. These cyclic procedures of evaluation-selection-application of genetic operators continue over generations until the evaluation result meets a pre-defined stop condition. In a pseudo code (Buckles and Petry, 1992), the principle underlying a genetic algorithm can be expressed as follows:

```
begin
   initialize population
   evaluate initial population
   while not (stop condition) do
   begin
      select organisms
      select a genetic operator
      generate new population by applying a genetic operator
      evaluate newly generated population
   end
end
```

This structure is common to all genetic algorithms. The specific challenge in any application domain concerns the specification of a representation scheme of the possible solutions, the specification of the fitness function, the choice of genetic operators, and the choice of stop condition. Note that each generation selects and applies only one genetic operator. Equally important is a list of parameters, to be set by the researcher, that controls the overall algorithmic flow. It includes the number of generations, the size of a population, the selection probabilities of genetic operators in each generation, the number of organisms to be selected from a population, the probabilities of reproduction and crossover to each selected organism, and the mutation probability of each gene of the selected organism. We will develop a Genetic Algorithms-based MultiDimensional Sequence Alignment Method (GA-MDSAM) by incorporating some context-specific modifications of these principles.

3.1.1 Representation scheme: The terms underlying GA are used as follows for our problem at hand:

generation → generation
population → population
organism → trajectory set (consisting of K uni-dimensional trajectories)
chromosome → trajectory
gene → move (vertical, horizontal and diagonal)

where: a set of moves in the comparison table constitutes a trajectory of alignments of an attribute; a set of trajectories composes a trajectory set as a solution of multidimensional alignments that will be converted into a multidimensional alignment cost; a set of trajectory sets constitutes a population of the current generation.

The resulting representation of a population can then be denoted as:

$$E(t) = \{E_1, \ldots, E_u, \ldots, E_U\} \tag{7}$$

with

$$E_u = [E_{1u} \ldots E_{ku} \ldots E_{Ku}]' \tag{8}$$

and

$$E_{ku} = [X_{1k} \ldots X_{lk} \ldots X_{(m+n)k}] \tag{9}$$

where,

$E(t)$ is the population of the t-th generation ($t \geq 0$);
E_u is the u-th trajectory set of $E(t)$ and is an ordered set of K trajectories ($K > 0$);
U is the number of trajectory sets in $E(t)$ ($0 < U << TU$);
E_{ku} is the k-th trajectory of E_u and is an ordered set of moves of the comparison table;
X_{lk} is the l-th move of the k-th trajectory and is encoded as `V`, `H` or `D` respectively representing the vertical, horizontal or diagonal move.

It is important to note that when two K-dimensional activity patterns of lengths m and n are compared, the number of operations is fixed and equal to $m+n$ for all trajectories. In other words, GA-MDSAM employs a 'fixed-format' representation scheme although the lengths of actual trajectories in the comparison tables may vary because the lengths of the trajectories having diagonal moves of identity operation are shorter than those of off-diagonal moves. This representation scheme intends to separate population generation procedures (initialization and genetic operator application) from evaluation procedures. Any comparison of two sequences must yield a trajectory that ends with the last pair of source and target elements (m and n) in the comparison table. However, a randomly generated fixed-format trajectory may not satisfy this condition. The population generation procedures do not check the end condition when generating trajectory sets because this non-supervised search can improve the results in later generations by not narrowing the search space and keeping the diversity of the solutions even though it may decrease the efficiency, particularly in early generations. In fact, some preliminary experimentation demonstrated that a supervised search strategy is not better than the non-supervised search strategy.

***3.1.2 Fitness function and trajectory-set selection*:** GA-MDSAM assumes that the trajectory sets with lower multidimensional alignment costs are the better ones and hence, takes the best-of-generation solution (Koza, 1992) as the fitness of each generation. The fitness function, also called objective function, is therefore defined by equations (3) - (6) in section 2.1. In particular, each trajectory E_{ku} of E_u of the t-th generation, consisting of vertical, horizontal and diagonal

moves, is converted into the corresponding uni-dimensional operation set that contains deletion and insertion operations. A substitution is automatically decomposed into a deletion and an insertion, as mentioned in section 2.1. The uni-dimensional operation sets are then integrated into multidimensional operation sets by equation (6). The fitness values C^u of individual trajectory sets E_u are determined by equations (4) and (5). Finally, the fitness of population C^* at the t-th generation is then determined by equation (3). Note that the number of multidimensional operation sets of GA-MDSAM (U) of each generation is however a much smaller one than that of the exhaustive MDSAM (TU) expressed in equation (3).

Now the problem is how to evaluate particular trajectory sets whose uni-dimensional trajectori(es) do not end with the element pair (m,n). We will handle this problem by simply ignoring the moves that do not satisfy this requirement and, if necessary, adding alternative moves. Consider, for example, a trajectory of alignments, involving two sequences of lengths 3 and 4 (Fig. 3). The trajectory resulting from a population generation procedure is E_{ku} = [D V H V D V V] in the comparison table of the LHS of Fig. 3, where n(E_{ku}) = 3+4 = 7. The underlined elements in the trajectory vector are the moves not in the comparison table, which hence, are ignored. Instead, new moves are added for evaluation in the RHS of Fig. 3, and the resulting trajectory is assumed to be E_{ku} = [D V H V H H]. Note that the trajectory itself is not changed, but the evaluation procedure assumes an imaginary trajectory only for evaluation purposes. The generated trajectory is unchanged even if part of it falls outside the comparison table, and will be input for the population generation procedure of the next generation. This is because GA-MDSAM employs a non-supervised search strategy that strictly separates the population generation from evaluation procedures to maintain the diversity of trajectory-set candidates. In addition, unlike the vertical and horizontal moves that are always converted into deletion and insertion operations, it is checked whether the diagonal move implies identity (s_i=g_j) or substitution (s_i≠g_j), and the move is decomposed into vertical and horizontal moves if it is known as substitution.

Trajectory sets for generating new populations are selected probabilistically, based on their fitness values. This so-called roulette wheel selection scheme (Goldberg, 1989), is known to result in better solutions, and can be expressed as:

$$S(E_u) = \frac{\sum_{u'}^{U} C^{u'}}{\sum_{u'}^{U} \frac{C^u}{C^{u'}}} \qquad (10)$$

where,

$S(E_u)$ is the selection probability, also called survival rate, of the u-th trajectory set.

Note that equation (10) holds for *each* selection of a trajectory set in the current generation. In other words, the selection of trajectory sets is made with replacement, and gives higher and higher chances to better solutions to be selected.

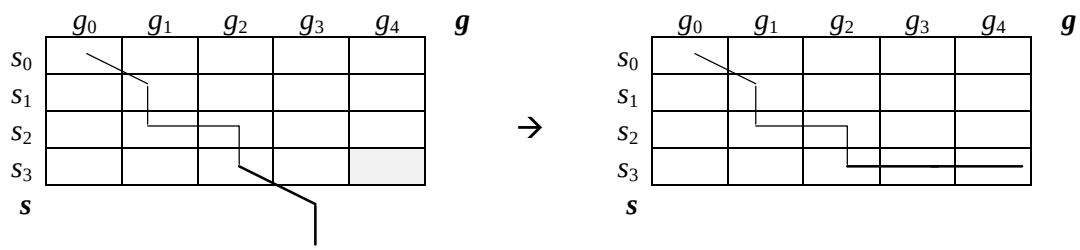

Fig. 3: An 'illegal' trajectory and a correction of it for the evaluation

***3.1.3 Genetic operators*:** GA-MDSAM employs reproduction, crossover and mutation operators, commonly used in GA. It is well known however that these genetic operators may vary considerably, depending on the data. For example, Syswerda (1991) argued that "constructing operators can be tricky, and experimentation is often necessary to get them right. The chromosome should be broken up in a way that is "natural" for the problem at hand." We therefore explored several combinations of genetic operators of different forms to identify the best form for our problem.

First, we explored three different kinds of crossover operators: one-point crossover, two-point crossover and uniform crossover, as illustrated in Fig. 4. One-point crossover is an operator that exchanges one piece of information between two selected trajectories, designated by a single cutting point somewhere in the sequence. Two-point crossover is an operator that exchanges one or two pieces of information between two selected trajectories, designated by two cutting points somewhere in the sequence. Uniform crossover is an operator that exchanges the information of each corresponding position between two selected trajectories by chance. It is a crossover operator, possibly involving multiple cutting points (Davis, 1991b).

Secondly, we explored two kinds of mutation operators: point mutation and order-based mutation. Point mutation is an operator that changes by chance the kind of a move. Order-based mutation (Syswerda, 1991) is an operator that randomly selects two moves and exchanges them within a single trajectory.

Finally, we explored the reproduction operator as it is commonly used in conventional GA. Once a selection of trajectory sets has been made on the basis of fitness values, the reproduction operator simply copies the selected trajectory sets to the next generation.

***3.1.4 Stop condition*:** In general, three different stop conditions can be identified. First, the number of generations for a run is often a priori given (Koza, 1992). Secondly, a certain threshold value of the fitness of an organism may be employed. Finally, a certain degree of stability or

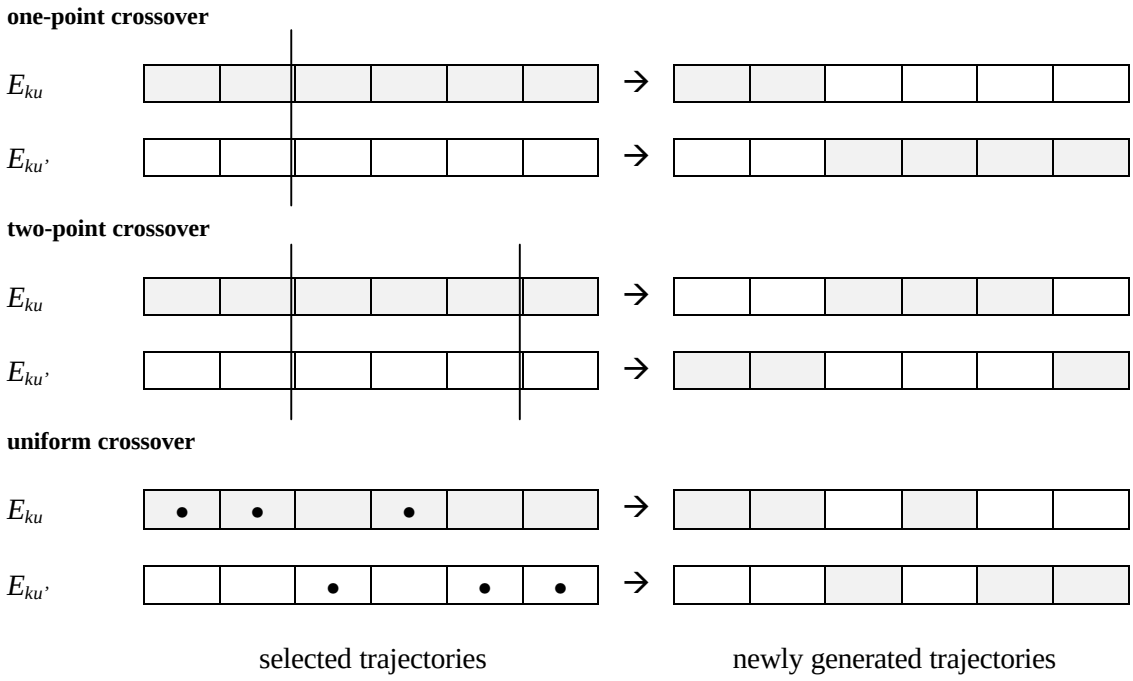

Fig. 4: Illustration of different crossover operators. The vertical lines represent crossover points; each cell represents a vertical, horizontal or diagonal move. Note that the number of cells (moves) of a

trajectory is $m+n$, and that that each crossover event works on a pair of trajectories for the same attribute. That is, $k = k'$ in any pair of E_{ku} and $E_{k'u'}$ where $u \neq u'$.
convergence rate across consecutive generations can be specified. We decided to use a certain number of generations as the stop condition. The exact number was based on some preliminary experiments that we conducted.

3.1.5 Parameters: Three additional parameters are particularly relevant. First, we specified the choice probability of the diagonal move in initializing the population and in mutating trajectory sets. Because we have vertical, horizontal and diagonal moves, a choice probability of 1/3 given to every move may be considered reasonable. However, many of diagonal moves involve substitution rather than identity, and the equal choice probability rule is not fair for the problem at hand. We therefore introduced an additional parameter of diagonal move choice that was assigned a value greater than 1/3.

Secondly, we defined a parameter to select the best trajectory set(s) for the next generation. The best trajectory set of the current generation contains all the efforts accumulated from the first generation. When the best combination(s) of a generation is not selected, the algorithm may put in the same effort again and again over generations. To prevent this inefficiency, we introduced a second additional parameter, which defines the number of best trajectory sets to be preserved for the next generation, instead of being selected by chance.

Finally, we were concerned about the possible limitation of conventional genetic operators with respect to their ability to derive better trajectory sets from the selected ones. In our application context, the genetic operations with fixed format representation may produce a trajectory set that is worse than the selected one. To increase the possibility to find a locally better trajectory set, we generated several offspring around the selected trajectory set and picked the best. This nearest neighbor heuristic selects the locally superior alternatives at each search step (Rich and Knight, 1991; Reeves and Höhn, 1996). Of course, not all neighbors of a selected trajectory set were tried as all possible crossover or mutation of a trajectory set would cost too much computing time. Instead, we examined a few neighbors by introducing a parameter for the number of neighbors.

In summary, GA-MDSAM can now be detailed as follows:

```
begin
   t = 0                                   // t indicates the t-th generation //
   initialize E(t)
   calculate C*(t)
   Best_Fitness = C*(t)
   while not (t = T) do
   begin
      select a genetic operator
      create E1(t) by selecting and copying a subset of E(t)
      t = t+1
      create E1(t) by applying the selected genetic operator to E1(t–1)
      create E2(t) by selecting and copying a subset of E(t–1)
      create E(t) by summing E1(t) and E2(t)
      calculate C*(t)
      if C*(t) < Best_Fitness then Best_Fitness = C*(t)
   end
end
```

where $C^*(t)$ denotes the fitness of $E(t)$.

Note that $E_1(t)$ consists of the trajectory sets newly generated by the single or multiple applications of a genetic operator to each of the selected trajectory sets and that $E_2(t)$ includes the best trajectory set(s) of the (t−1)-th generation.

3.2 A heuristic method based on dynamic programming

GA-MDSAM separates the solution search procedure from the solution evaluation procedure and keeps the solution search context-free. This non-supervised, general-purpose search scheme may find better solutions by avoiding the premature application of knowledge about uni-dimensional alignments to the integration of operation sets. In theory, the uni-dimensional optimum alignments are not necessarily an appropriate basis for the multidimensional integration of operation sets in terms of the optimality of the solutions, as we have discussed in section 2.2. In practice, as the uni-dimensional optimum trajectories involve the largest number of cost-free identity operations, we may expect that their integration will result in a solution that is near to the multidimensional optimum. However, there are often many possible optimum trajectories as mentioned in section 2.1, and the number of combinations of trajectories across attributes to consider may become intractably large. As tracing all possible uni-dimensional optimum trajectories would cause the problem of combinatorial explosion, we therefore suggest employing a more simplified heuristic approach that considers, for each attribute dimension, only one optimum trajectory along the diagonal region of the comparison table. The rationale underlying this heuristic approach lies in the expectation that the integration of diagonal-oriented optimum trajectories across attribute dimensions will also lead to a good solution, as most uni-dimensional optimum trajectories run along the diagonal region of the comparison table (Kruskal and Sankoff, 1983). More similar trajectories of different attribute dimensions have more similar operation sets, and more similar operation sets of different attribute dimensions result in more occasions of operation integration in the multidimensional space. Examples of the diagonal region of the comparison table are illustrated as shadowed cells in Fig. 5. Among the uni-dimensional optimum trajectories, the optimum trajectory along the diagonal region is the one that has the largest number of identity operations whose coordinates fall in the diagonal region.

Equations (3) - (6) in section 2.1 representing the general solution of the MDSAM problem involved the existence of multiple uni-dimensional operation sets and hence, assumed the existence of multiple multidimensional operation sets. Although reducing the number of multidimensional operation sets, GA-MDSAM searches multiple opportunities for the solution because it also involves multiple uni-dimensional operation sets. In contrast, the proposed Diagonal-directed Optimum-Trajectory-based MultiDimensional Sequence Alignment Method (DOT-MDSAM) considers only one multidimensional operation set, as it is involved with only one uni-dimensional operation set for the operation integration. Consequently, the proposed DOT-MDSAM can also be expressed by equations (3) - (6) in section 2.1, in which the number of multidimensional operation sets, *TU*, is assumed to be 1. The only uni-dimensional operation set of each attribute dimension by which equation (6) induces a single multidimensional operation set is then defined as:

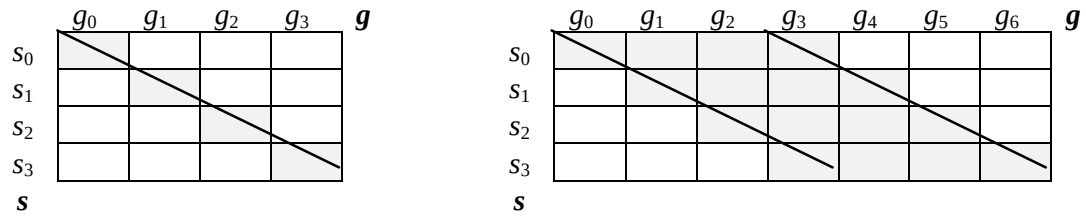

Fig. 5: Diagonal region of the comparison table

$$O_k = \{p | p = \mathsf{d}(i,k) \vee \mathsf{i}(j,k)\} \tag{11}$$

with

$$O_k = conv(Q_k) \tag{12}$$

and

$$Q_k = \{q | q = \mathsf{e}(i,j,k)_1, \ldots, \mathsf{e}(i,j,k)_r, \ldots, \mathsf{e}(i,j,k)_{R_k}\} \tag{13}$$

and

$$F(Q_k) = F(Q_{v_k^*}) = \max[F(Q_{1_k}), \ldots, F(Q_{v_k}), \ldots, F(Q_{V_k})] \tag{14}$$

and

$$F(Q_{v_k}) = \sum_{r}^{R_k} e_{r_v} \tag{15}$$

and

$$e_{r_v} = \begin{cases} 1 & \text{if } \mathsf{e}(i,j,k)_{r_v} \in D\{\mathsf{e}(i,j,k)\} \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

where,

O_k is the cost-taking deletion and insertion operation set of the k-th attribute dimension;

Q_k is the cost-free identity operation set of an optimum alignment of the k-th attribute dimension;

$conv(Q_k)$ is a procedural function that converts Q_k into O_k;

$\mathsf{e}(i,j,k)_r$ is the r-th identity operation applied to the i-th source element and the j-th target element of the k-th attribute dimension;

R_k is the number of identity operations of the optimum trajectory of the k-th attribute dimension;

Q_{v_k} is the identity operation set of the v-th optimum trajectory of the k-th attribute dimension;

V_k is the number of optimum trajectories that can be traced in the comparison table of the k-th attribute dimension;

$F(Q_{v_k})$ is a 'diagonal' function measuring how much Q_{v_k} is involved with the diagonal region of the k-th attribute dimension, denoted by $D\{\mathsf{e}(i,j,k)\}$;

e_{r_v} is a dichotomous value denoting whether the coordinate of the r-th identity operation of the v-th optimum trajectory falls in the diagonal region;

$\mathsf{e}(i,j,k)_{r_v}$ is $\mathsf{e}(i,j,k)_r$ of the v-th optimum trajectory.

More specifically, the value of e_{r_v} in equation (16) is determined as:

$$e_{r_v} = \begin{cases} 1 & \text{if } p \le q \le (|m-n|+p) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where p and q are the positions of the shorter pattern and the longer pattern appeared in $e(i, j, k)_{r_v}$, respectively.

Converting Q_k into O_k can be easily done by comparing the coordinates of two adjacent identity operations. All the source and target elements in-between these are identified as the elements that are respectively deleted and inserted. When there are multiple Q_{v_k}'s, with the maximum value of the 'diagonal' function, one of such Q_{v_k}'s is arbitrarily selected as $Q_{v_k^*}$.

4. Illustration

The illustration compares the MDSAMs using the two different heuristics developed in section 3. The comparison is conducted in terms of how close the results of each heuristic are to the 'true' similarities, and in terms of the computing time required to reach the solution. More specifically, because different combinations of parameters yield different GA-MDSAMs, we compare the DOT-MDSAM and a set of GA-MDSAMs. GA-MDSAMs can be differentiated in terms of the kinds of crossover and mutation operators, and the number of neighbors of each selected trajectory sets. According to these criteria, twelve GA-MDSAMs were compared (Table 1). Table 2 gives a summary of some other operational decisions. Based on preliminary experiments, the analyses were restricted to 400 generations and 100 trajectory sets. The probability of diagonal move choice was given a higher probability than the probability of a horizontal or vertical move. Likewise, the probability of crossover selection was set higher than the probability of either a reproduction or a mutation.

We conducted 2485 pairwise comparisons of 71 three-dimensional activity patterns consisting of activity type, location and transport mode sequences. As the exhaustive search requires considerable amount of computing time, only this small number of activity patterns was analyzed. The set of 71 activity patterns was arbitrarily selected from 2974 activity patterns collected in two cities (Hendrik-Ido-Ambacht and Zwijndrecht) in The Netherlands. The activity patterns distinguish forty-eight activity types and five transport modes (car, walk, bike, car passenger and public transport mode including bus, train, tram, and subway). Activity locations are encoded by four-digit zip codes, except for the home location that is encoded '0'. In addition, the analysis in this section includes an extra code representing the attribute elements that were not identified. The substitution operation was applied to the alignment of the unknown element. The analysis also included an extra activity type, 'in-home activities.' This activity type was identified if pre-specified thirteen particular activity types among forty-eight activity types were conducted at home. Once identified, each set of consecutive in-home activities was encoded as a single in-home activity. The average length of the 71 activity patterns that were adjusted in this way was 6.80 with a maximum of 17 and a minimum of 1 activities, while the average of the 2974 activity patterns was 6.78 with a maximum of 21 and a minimum of 1.

Table 1. GA-MDSAMs to test

	one-point crossover		two-point crossover		uniform crossover	
	point mutation	order-based mutation	point mutation	order-based mutation	point mutation	order-based mutation
1 neighbor	(1-1-1)	(1-2-1)	(2-1-1)	(2-2-1)	(3-1-1)	(3-2-1)
5 neighbors	(1-1-5)	(1-2-5)	(2-1-5)	(2-2-5)	(3-1-5)	(3-2-5)

Note: The first numeral in the bracket represents the kind of crossover: 1=one-point, 2=two-point and 3=uniform. The second numeral denotes the kind of mutation: 1=point, 2=order-based. The last numeral indicates the number of neighbor trajectory sets to try.

Table 2. Parameters commonly used for twelve GA-MDSAMs

parameter	value	parameter	value
# generations	400	trajectory set proportion for reproduction	30 %
# trajectory sets	100		
# best trajectory sets	1	trajectory set proportion for crossover	40 %
vertical move choice prob.	20 %		
horizontal move choice prob.	20 %	trajectory set proportion for mutation	20 %
diagonal move choice prob.	60 %		
reproduction selection prob.	20 %	reproduction application prob.	100 %
crossover selection prob.	70 %	crossover application prob.	100 %
mutation selection prob.	10 %	mutation application prob.	1 %

Table 3 provides the overall results of the comparison of DOT-MDSAM and the GA-MDSAMs. It is clear from the table that DOT-MDSAM outperforms GA-MDSAMs in both terms of costs and computing time. DOT-MDSAM yields 2018 correct results (81.2 % of total 2485 pairwise comparisons) in just 11 seconds. In contrast, on average, the percentage of correct predictions of GA-MDSAMs is only 73.1 %. It implies that most of the different combinations of uni-dimensional trajectories induce the same set of operations in the multidimensional space and that DOT-MDSAM therefore likely provides the true similarities or at least the similarities that are very close to the true ones within a short time. In comparison with DOT-MDSAM, even the best of GA-MDSAMs yields only 1938 correct results (78 % of the total), while consuming much more computing time (more than two hours).

Interestingly, there are some notable differences in performance between GA-MDSAMs. The proportion of correct results, the average deviation from the true similarities and the computing time all vary considerably between GA-MDSAMs. Table 4 compares GA-MDSAMs in terms of the kind of crossover operator, the kind of mutation operator and the

Table 3. Performances of DOT-MDSAM and GA-MDSAMs (2485 comparisons for each measure)

MDSAMs		=[I]	range[II]	aver diff[III]	computing time[IV]
exhaustive search		-	-	-	62hrs 47min 32sec
DOT-MDSAM		2018 (81.2%)	0 - 3	.232	11sec
GA-MDSAMs	(1-1-1)	1776 (71.5%)	0 - 7	.596	56min 13sec
	(1-1-5)	1879 (75.6%)	0 - 6	.480	1hr 52min 42sec
	(1-2-1)	1729 (69.6%)	*0 - 10*	.641	***55min 31sec***
	(1-2-5)	1833 (73.8%)	0 - 6	.516	1hr 49min 42sec
	(2-1-1)	1751 (70.5%)	0 - 6	.602	56min 40sec
	(2-1-5)	1862 (74.9%)	0 - 7	.496	1hr 54min 17sec
	(2-2-1)	*1703* (68.5%)	0 - 7	*.662*	55min 49sec
	(2-2-5)	1818 (73.2%)	0 - 6	.542	1hr 51min 14sec
	(3-1-1)	1826 (73.5%)	0 - 6	.530	59min 54sec
	(3-1-5)	***1938*** (78.0%)	***0 - 4***	***.404***	*2hrs 11min 6sec*
	(3-2-1)	1767 (71.1%)	0 - 6	.575	59min 9sec
	(3-2-5)	1909 (76.8%)	0 - 5	.433	2hrs 8min 6sec
	best	1938 (78.0%)	0 - 4	.404	55min 31sec
	worst	1703 (68.5%)	0 - 10	.662	2hrs 11min 6sec
	average	1816 (73.1%)	6.33	.540	1hr 26min 42sec

I: Heuristic MDSAM costs are the same as the true costs obtained by the exhaustive search.
II: Range of deviation of each measure's results from the true similarities.
III: Average deviation from the true similarities.
IV: Computations based on a 350MHz Pentium II processor.

Table 4. Performances of GA-MDSAMs by different parameters

parameter	(value)	<	computing time
crossover	one-point	680.75 (27.39%)	1hr 23min 32sec
	two-point	701.50 (28.23%)	1hr 24min 12sec
	uniform	625.00 (25.15%)	1hr 34min 34sec
mutation	point	646.33 (26.01%)	1hr 28min 29sec
	order-based	691.83 (27.84%)	1hr 26min 35sec
# neighbors	1	726.33 (29.23%)	57min 13sec
	5	611.83 (24.62%)	1hr 57min 52sec

Note: The figures are averaged on each of the concerned parameters

number of neighbor trajectory sets. Table 4 indicates that among the crossover operators, uniform crossover yields better results although it consumes slightly more time. There appears to be no clear difference however between the mutation operators, which might reflect that only a few mutations have taken place. Table 4 also demonstrates that multiple neighbor searches perform much better than the single neighbor search of conventional GA at the expense of a considerably more computing time.

Yet, the point mutation and multiple neighbor searches are worse than DOT-MDSAM in terms of the correctness of the results and computing time. It is therefore safe to say that DOT-MDSAM, a heuristic method that utilizes context-specific information of uni-dimensional sequence alignments, outperforms GA-MDSAM, a general-purpose search heuristic method that does not use context-specific information.

5. Conclusion and Discussion

This paper aimed at developing more efficient algorithms to measure the similarities between multidimensional activity patterns within acceptable computer times by utilizing heuristics that appropriately reduce the multidimensional solution search space. In particular, the paper developed two alternative multidimensional sequence alignment methods. The first employed genetic algorithms. The second alternative was developed on the basis of a dynamic programming technique. The major difference between the two alternatives is that the latter utilizes context-specific information in producing the solution, while the former does not rely on such information.

The illustration compared the performances of the suggested methods. It clearly showed that the heuristic utilizing the dynamic programming technique outperforms the genetic algorithms-based heuristic, both in terms of the correctness of the results and computing time. In other words, the dynamic programming method reduces the search space very drastically and effectively so that, in our application context, it outperforms even the most powerful heuristic of blind search method, which GA probably is.

Based on these observations, we conclude that the future research should focus on some further elaboration of multidimensional sequence alignment methods based on the dynamic programming heuristic. First, we assumed that the weighting value of a substitution is the simple sum of deletion and insertion weighting values. Although this assumption is not problematic in the present context, a generalized method with a flexible substitution weight is still required. Secondly, in this paper we considered only categorical attributes. The problem of handling quantitative attributes has to be solved given the importance of activity duration, which is inherently quantitative in nature. Finally, we selected a uni-dimensional trajectory arbitrarily among the optimum trajectories in the diagonal region of the comparison table. We may however further improve the results of the suggested heuristics by systematically comparing all optimum trajectories in the diagonal region. This would not require that much

extra computing time as the number of optimum trajectories may be relatively small, compared to the number of entire optimum trajectories.

References

Arentze, T. A., F. Hofman, C. H. Joh and H. J. P. Timmermans, 1998, Experiences with developing ALBATROSS: A Learning-Based Transportation Oriented Simulation System, in K. J. Beckmann (ed.), *Verkehr und Mobilität*, Aachen: Institut für Stadtbauwesen, Stadt Region Land 66, 61-70.

Buckles, B. P. and F. E. Petry, 1992, An overview of genetic algorithms and their applications, in B. P. Buckles and F. E. Petry, *Genetic Algorithms*, Washington: IEEE Computer Society Press, 1-4.

Burnett P. and S. Hanson, 1982, The analysis of travel as an example of complex human behavior in spatially-constrained situations: definition and measurement issues, *Transportation Research* A, 16, 87-102.

Davis, L., 1991a, Performance enhancements, in L. Davis (ed.), *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold, 23-42.

Davis, L., 1991b, Further evolution of the genetic algorithm, in L. Davis (ed.), *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold, 43-53.

Ettema, D. F. and H. J. P. Timmermans, 1997, Theories and models of activity patterns, in D. F. Ettema and H. J. P. Timmermans (eds.), *Activity-Based Approaches to Travel Analysis*, Oxford: Pergamon, 1-36.

Gärling, T., R. Gillholm, J. Romanus and M. Selart, 1997, Interdependent activity and travel choices: behavioral principles of integration of choice outcomes, in D. F. Ettema and H. J. P. Timmermans (eds.), *Activity-Based Approaches to Travel Analysis*, Oxford: Pergamon, 135-149.

Goldberg, D. E., 1989, A gentle introduction to genetic algorithms, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Amsterdam: Addison-Wesley, 1-25.

Golob, T. F. and W. W. Recker, 1987, Dynamic analysis of complex travel behavior using a sub-sample of the Dutch National Mobility Panel, *Analyses of Panel Data: Proceedings of the Round Table Conference on the Longitudinal Travel Study*, The Hague: Projectbureau Integrale Verkeers- en Vervoersstudies, 173-193.

Gower, J. C., 1971, A general coefficient of similarity and some of its properties, *Biometrics*, 27, 857-871.

Joh, C. H., T. A. Arentze, F. Hofman and H. J. P. Timmermans, 1997, Activity pattern similarity: towards a multidimensional sequence alignment, *Paper presented at the IATBR Meetings in Austin, Texas, U.S.A.*

Koza, J. R., 1992, *Genetic Programming: On the Programming of Computers by means of Natural Selection*, London: MIT Press.

Kruskal, J. B., 1983, An overview of sequence comparison, in D. Sankoff and J. B. Kruskal (eds.), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, London: Addison-Wesley, 1-44.

Kruskal, J. B. and D. Sankof, 1983, An analogy of algorithms and concepts for sequence comparison, in D. Sankoff and J. B. Kruskal (eds.), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, London: Addison-Wesley, 265-310.

Pas, E. I., 1983, A flexible and integrated methodology for analytical classification of daily travel-activity behavior, *Transportation Science*, 17, 405-429.

Reeves, C. R. and C. Höhn, 1996, Integrating local search into genetic algorithms, in V. J. Rayward-Smith, I. H. Osman, C. R. Reeves and G. D. Smith (eds.), *Modern Heuristic Search Methods*, Chichester: John Wiley & Sons, 99-115.

Recker, W. W., M. G. McNally and G. S. Root, 1985, Travel/activity analysis: pattern recognition, classification and interpretation, *Transportation Research* A, 19(4), 279-296.

Rich, E. and K. Knight, 1991, Problems, problem spaces, and search, in E. Rich and K. Knight, *Artificial Intelligence*, London: McGraw-Hill, 29-62.

Syswerda, G., 1991, Schedule optimization using genetic algorithms, in L. Davis (ed.), *Handbook of Genetic Algorithms*, New York: Van Nostrand Reinhold, 332-349.

Young, T. Y. and T. W. Calvert, 1974, Image analysis and character recognition, in T. Y. Young and T. W. Calvert, *Classification, Estimation and Pattern Recognition*, New York: American Elsevier, 311-354.

Wilson, C., 1998, Activity pattern analysis by means of sequence-alignment methods, *Environment and Planning* A, 30, 1017-1038.