DBA and K-means clustering : Explainability problems for research on behaviors and strategies

Axel Palaude, Thierry Viéville

DBA and K-Means clustering : Explainability problem for strategy and behavioral research

Axel Palaude, Thierry Viéville

ISSN 0249-6399 ISRN INRIA/RT--9522--FR+EN

RESEARCH REPORT

N° 9522

December 2023

MNEMOSYNE

DBA and K-means clustering : Explainability problems for research on behaviors and strategies

Axel Palaude[1], Thierry Viéville[2]
MNEMOSYNE

Research Report N° 9522— December 2023 —16 pages.

Abstract: Studying creative problem solving (CPS) activities can enrich our understanding of human learning by facilitating the observation of behaviors related to realistic situations of complex open-ended problem solving . Actions taken by an individual in such activities are situated in time, thus allowing the creation of time sequences of observations. These observations are not purely numerical, and thus can be represented by symbolic time sequences.
In particular, the formal analysis of such time sequences leads to a better understanding of the role of temporality in the adoption of CPS strategies.
For this, we propose to apply clustering algorithms in order to group sequences in categories that could be associated with emergent behavioral patterns, either at the whole activity level or at the level of subsequences of actions. Clustering requires the definition of a distance between symbolic data structures representing observations, used to determine the distance (more precisely dissimilarity), and the capacity to interpret such clusters.
Our ongoing research uses Dynamic Time Warping (DTW) as an algorithm to compute dissimilarity between sequences. This technical report addresses the issue of computing a K-means clustering algorithm and centroids with DTW for symbolic sequences, using an approximation algorithm named DTW Barycenter Averaging (DBA).
It appears that DBA for symbolic sequences is not relevant as it produces non-explainable sequences for the pattern analysis, but this negative result is of interest and allows us to propose an alternative.
Clustering algorithms using medoids will be preferred in future research, and we explain why in this report.

Key-words: creative problem-solving, learning analytics, dynamic time warping, clustering.

[1] Axel Palaude, Inria Mnemosyne Team, AIdE – axel.palaude@inria.fr

[2] Thierry Viéville, Inria Mnemosyne Team, UCA LINE Laboratory, thierry.vieville@inria.fr

RESEARCH CENTRE
BORDEAUX - SUD-OUEST

351 Cours de la Libération
Bâtiment A29
33405 Talence Cedex France

DBA et le clustering des k-moyennes : Des problèmes d'explicabilité pour la recherche sur les stratégies et comportements

Axel Palaude[3], Thierry Viéville[4]
MNEMOSYNE

Rapport de Recherche N° 9522—December 2023—16 pages

Résumé : L'étude des activités de résolution créative de problèmes enrichit notre compréhension de l'apprentissage humain en permettant d'observer les comportements humains dans des situations réelles de résolution de problèmes ouverts et complexes. L'observation des actions effectuées est située dans le temps, ce qui permet la création de séquences temporelles d'observations, qui peuvent être représentées par des séquences symboliques. L'analyse de ces séquences temporelles permet ainsi de mieux comprendre le rôle de la temporalité dans l'adoption de stratégies pour la résolution créative de problèmes. Pour cela, nous proposons d'utiliser des algorithmes de clustering afin de regrouper des séquences similaires qui pourraient ainsi être associés à des types de comportements, que ce soit des comportements généraux pour l'activité ou des comportements relatifs à seulement une sous-séquence d'actions. Les algorithmes de clustering nécessitent la définition d'une distance entre ces structures temporelles et symboliques. Nous utilisons le Dynamic Time Warping (DTW) (ou déformation temporelle dynamique) comme méthode permettant de calculer la dissimilarité entre les séquences. Cependant, dans l'optique d'utiliser des algorithmes tels que l'algorithme des k-moyennes, nous avons besoin de mettre en place un algorithme permettant de calculer des centroïdes. Pour cela, nous utilisons un algorithme nommé le DTW Barycenter Averaging (DBA).
Cependant, le DBA n'est finalement pas pertinent dans notre analyse de données, car la méthode donne lieu à des centroïdes non interprétables, et pour d'autres raisons relatives à la structure hiérarchique de nos données symboliques. Ainsi, nos prochains travaux se concentreront sur l'utilisation des médoïdes à la place des centroïdes.

Mots clés : Résolution créative de problèmes, analyse de l'apprentissage, déformation temporelle dynamique, clustering

[3] Axel Palaude, équipe MNEMOSYNE (Inria), AIdE – axel.palaude@inria.fr

[4] Thierry Viéville, équipe MNEMOSYNE (Inria, Laboratoire LINE UCA, thierry.vieville@inria.fr

1. The CreaCube task modeling

1.1. The CreaCube task

Problem solving such as the Tower of Hanoi (Fansher et al., 2022) have been largely studied and permitted the formalization of general problem solving procedures using computational representations. In the case of the tower of Hanoi problem, (Newell & Simon, 1972), the representation is equivalent to a finite-state automaton (Glushkov, 1961), that is a 5-tuple consisting of :

- A *set of states* containing all possible states of the problem
- A *set of initial states* containing all possible initial states of the problem, in this case only one initial state.
- A *set of output states* containing all possible states that are considered as outputs, i.e. *final* states in which the problem is *solved*.
- An *input alphabet* consisting of all possible actions that it is possible to take (in this case, all actions consist of moving a disk from one tower to another, so the alphabet contains 6 possible actions)
- A *state-transition function* over the set of states and the input alphabet, that represents transitions between states depending on the action taken.

The representation of the automaton, as seen in Figure 1, is an example of well-defined problem-solving.

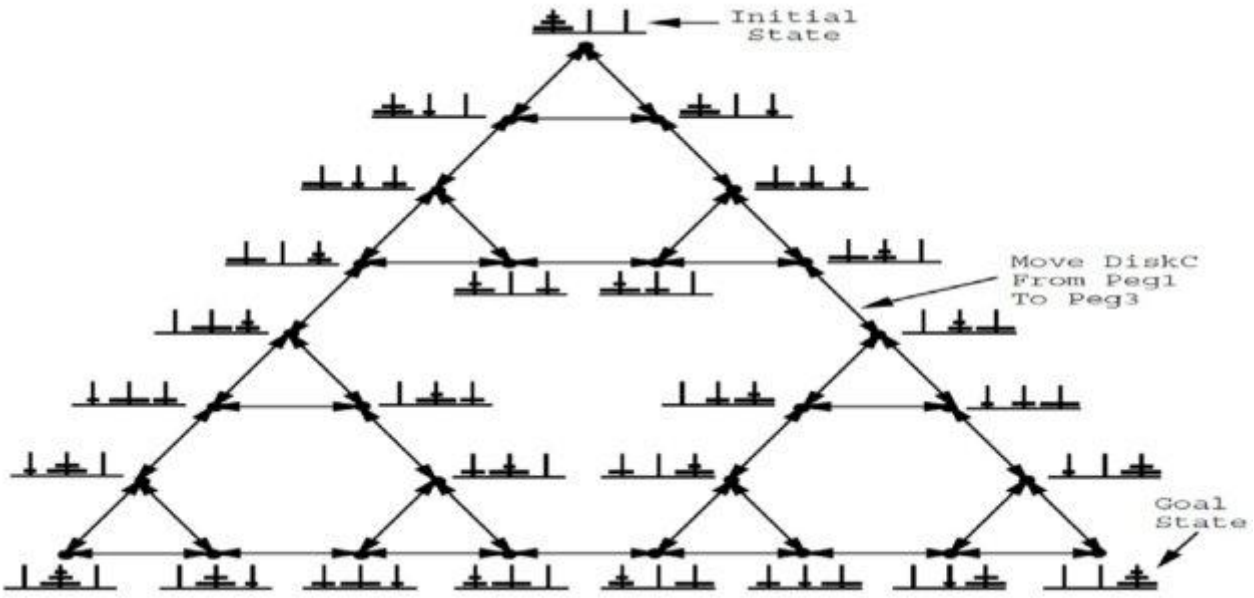

Figure 1 : The 3-disk Hanoi towers problem space, with one initial state and one output (goal) state with moving disks as a state-transition function, from (Knoblock, 1990)

We call this representation (states and transitions) the *problem space*, in which the subject is starting from an initial state and moving from state to state to reach an output state.

However, well-defined problems are not a good representation of more general problems, in which information can be unknown or obfuscated, and states and input alphabets can be virtually infinite, which is what we find in everyday problems or open problems at work.

These ill-defined problems are complex to model and require different levels of abstraction from the subject to solve it than a simple complete representation of it.

In authentic learning settings, problem solving is more often developed through complex and domain-specific problem-solving tasks (Molnár et al., 2013), including creative problem solving (CPS) tasks. CPS are ill-defined problem solving tasks that cannot be defined through a set of predefined states and require the learner to engage in an enactment of the conceptual or physical tools mediating the task. In ill-defined problems, there is no clear path to the set of output states, or even no clear set of output states. In such CPS tasks, the learner must build a representation of the problem, and then choose a behavior to adopt, which includes the choice of a (sub)goal and the problem space exploration (discovering parts of the problem space or moving through it). Actions undertaken by the learner in such tasks may be used as feedback to evaluate the result of the chosen behavior, evaluate the intermediate situation and decide to adapt (or not) the behavior to advance the task. Figure 2 represents a general representation of such problem space.

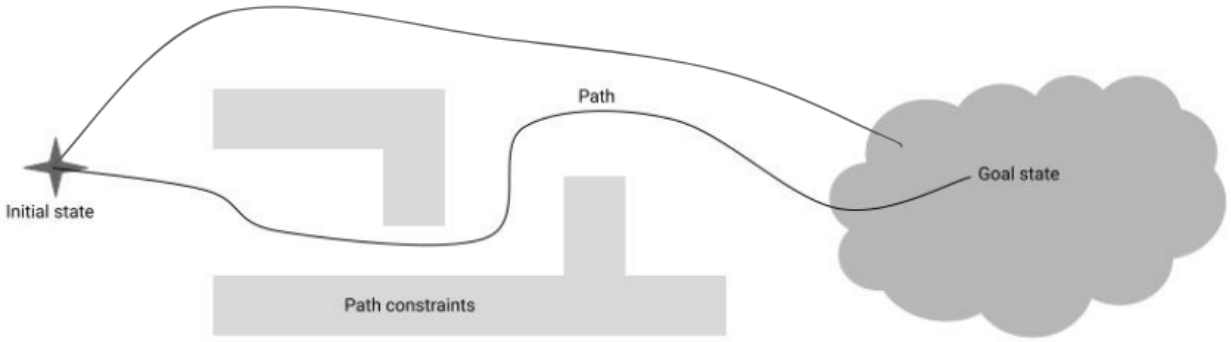

Figure 2 : A general representation of problem spaces. Paths represent sequences of actions taken by the learner in order to reach the goal state from the initial state. Path constraints cannot be passed through, which is why the learner needs to choose and adapt different behaviors in order to reach the goal state.

This study focuses on CPS tasks in which the subject needs to engage in a novel way of using a conceptual or physical tool which can satisfy the problem requirements. Our research goal is to characterize individuals' behaviors by identifying patterns across our corpus. This is done by using clustering methods that could be associated (or not) with general strategies for CPS. In particular, we are interested in an activity named Creacube.

CreaCube is an ill-defined robotic game-based activity (Leroy et al., 2021; Romero et al.,2018). In this CPS task, the participant is invited to manipulate four cubes of different colors. Each cube has different properties such as wheels or sensors, but these technological affordances are unknown to the subject when starting the activity. The goal of this CPS task is to build a vehicle composed of four pieces that can move autonomously from one point to another, without any further initial information about the problem.

Figure 3 : A configuration of the CreaCube activity

Four different cubelets (red, white, blue and black) are presented on a table (with a black point and a red point on it) to a player. The instructions, repeatable at will, consist in one sentence : « build a vehicle made up of four pieces that moves by itself from the red point to the black point ». The setup of the activity is shown in Figure 3. The black cube has a face with sensors, the blue cube has a switch on a face to activate a battery and the white

cube has wheels on a face that need to be alimented and receive instructions from sensors to move at a certain speed.

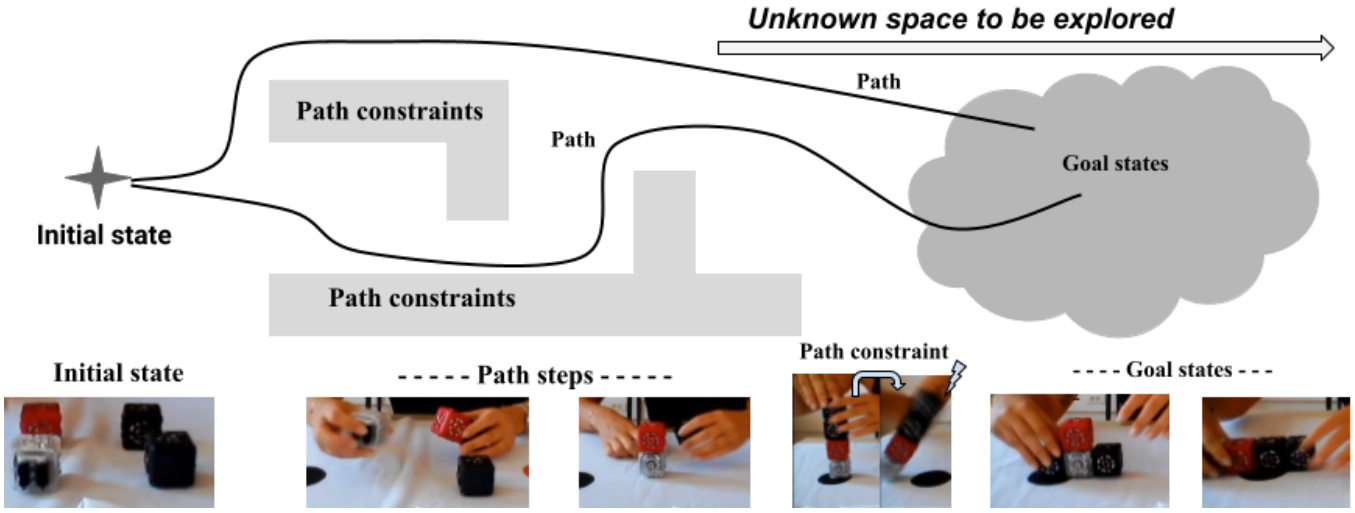

Figure 4 : CreaCube states associated to the general problem space representation

As for the problem space of the activity, we have an example (illustrated Figure 4) of the structure with one initial state and multiple goal states (i.e. structures of four that respect the conditions) with different path constraints corresponding to different types of problems (imbalance, unactivated battery etc.).

We can use the CreaCube activity in order to study some transversal competencies, also known as 21st-century skills, including problem solving, collaboration, creativity and computational thinking (*Romero, Lille et Patino*, 2017). The data collection of CreaCube experiments consists of sets of observables, and a set of observables represents the state of the activity at a given time. In this context, an observable represents a specific element of the scene : the actual configuration of cubes or the player's behavior. Observables require a decision on the way direct analysis of the situation or indirect one (through video analysis) or other data such learning analytics provided by the modular cubes can inform these observables. The set of observables used for this study is based on the work of (Mercier, 2022), as seen in Figure 5.

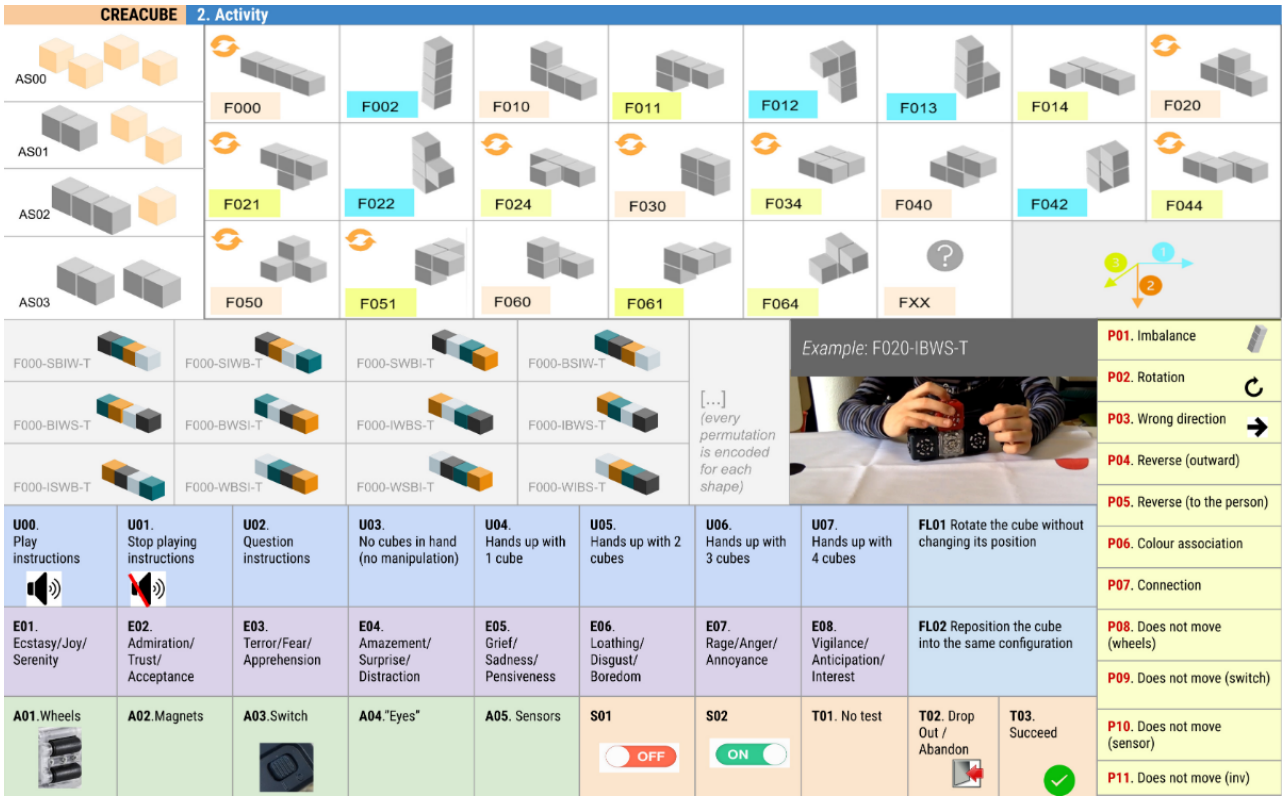

Figure 5 : Observables of the CreaCube activity, from (Mercier, 2022)

1.2. The data corpus

Based on a subset of the CreaCube corpus, selecting only children (7-12 y.o) who performed the task individually (93 subjects), we have for each experiment a data set including a time sequence starting when the subject hears the instructions for the first time and stopping when the subject succeeds or gives up. In each sequence, key events that can be observed have been annotated, resulting in 83 different types of observables annotated across all experiments. Detailed in Mercier (2022), they correspond to the discovery of affordances (the discovery of properties of some cubes), the realization of the most common four-cubes shapes, the criteria those shapes fulfill (or not) with regard to the guideline, and the final outcome of the activity. Those time sequences are thus symbolic trajectories, and we want to cluster them. However, it is difficult to create clusters of symbolic trajectories, mainly because it is difficult to define a measure of similarity between trajectories.

A trajectory of a subject is a list (sequence) of n-dimensonal boolean vectors, whose length is the length (in seconds) of the activity. Precisely, those vectors have 83 different dimensions corresponding to each observable. An observable can be activated or not at a given time, which is the reason why we are using boolean values.
In order to compare different trajectories, we need to compare similar elements from each. Those elements are the 83-dimensonal boolean vectors.

The first question to answer here is how to compare two vectors. We decided to use a simple edit distance : Considering that one edition is the change of a boolean value into another (either True to False or False to True) we can define the edit distance as the minimal number of editions necessary to transform one vector into another. The edit distance is common and can be defined in a lot of different symbolic environments. For instance, considering two words “call” and “salt”,we can define the edition as a letter change and in this case, the edit distance between those two words is 2 (c becomes s and l becomes t).
We will now give an example with our data : we consider two points in time in different activities. In the first one, the structure F002 is created, a tower of cube with only a problem of imbalance (the tower falls when the cubes move), and in the second one, another structure F020 is created with no other problem. The edit distance between those two points is 3 :

- The F002 observable is edited from True to False
- The Imbalance problem observable is edited from True to False

- The F020 observable is edited from False to True

Even with another distance, another problem arises : how to extend the distance between two points to a distance between two trajectories ? We can think of a naive approach consisting of simply adding distances between points at the same given time to compare two trajectories. For two trajectories, the distance will be the sum of the distance between the vectors of each trajectory at t = 0 s and between the vectors of each trajectory at t = 1 s, etc. However, this solution doesn't take into account the fact that trajectories are of different lengths, because subjects do not solve the problem at the same pace. It can be a matter of understanding or simply of speed of action. Adding empty vectors at the end of shorter trajectories doesn't solve the problem, as two subjects doing the exact same actions but at different paces will have a big distance value in total, wether during the comparison with "real vectors" who will be different at a given time or with "empty added vectors" who are different from non-empty vectors of the longer trajectory.

We need to use a method that allows a comparison between trajectories that erase time differences caused by e.g. pacing and to be adaptable depending on the chosen distance between vectors. This research report will concentrate on a method called Dynamic Time Warping (DTW) to define a similarity function and a method called DTW Barycenter Averaging (DBA) for prototype approximation of clusters.

2. Clustering symbolic data

2.1. Dynamic Time Warping on symbolic data

Dynamic Time Warping (DTW) is an algorithm used for measuring similarity between two temporal sequences which may vary in speed (Müller, 2007). The DTW method consists on finding an optimal match between two given time sequences by associating points from a time sequence to one or more points from the other sequences, with some restrictions :

- Every point from each sequence must be matched with at least one point from the other sequence
- The first and last points of one sequence are respectively associated with at least the first and the last points of the other sequence.
- Considering that the point i from one sequence is associated with the point j from the other sequence, the next point i+1 from the first sequence can only be associated with point j or points after j in the other time sequence, and vice versa.

The optimal match between two given time sequences is then the mapping of associated points that satisfies all of the previous restrictions and has the minimal cost of satisfying mappings, the minimal cost being the sum of absolute differences between values of each paired points.
Figure 6 presents an optimal matching between two time sequences with either the euclidean distance and the DTW optimal matching.

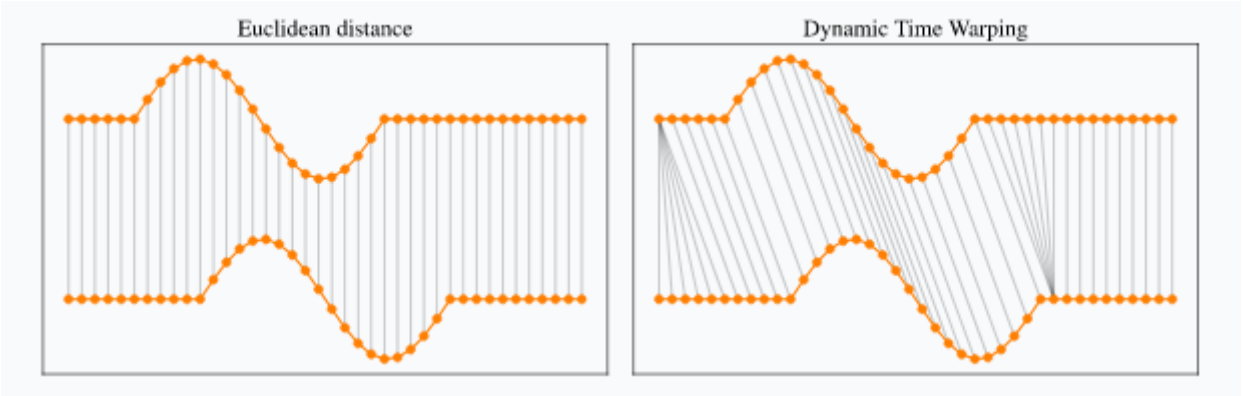

Figure 6 : Differences between euclidean distance and dynamic time warping (Tavenard, 2021)

The main benefit of using a dynamic matching algorithm like DTW is that it is able to take into account little variations between two similar sequences, for instance a temporal shift between two identical sequences. In particular, for Creacube, we consider that, as subjects are not identical, two subjects with the same behavior will have slightly different sequences, as there will be differences in, for instance, the execution speed.

We will now detail the algorithm for the dynamic time warping algorithm. DTW is based on the Levenshtein distance (Ney & Ortmanns, 1999). It is a dynamic algorithm that we compute using the following definition:

Given two sequences A and B :

$$A = a_1 a_2 ... a_n, B = b_1 b_2 ... b_m$$

We can compute the distance D :

$$D(A_i, B_j) = d(a_i, b_j) + min(D(A_{i-1}, B_{j-1}), D(A_i, B_{j-1}), D(A_{i-1}, B_j))$$

So the distance between A and B will be :

$$D(A, B) = D(A_n, B_m)$$

This requires the definition of the distance d between two points.
For the CreaCube observables set, in order to compute the distance, we define a point by its position in time, and by a set of boolean values : each boolean value corresponds to an observable, and is set to 1 if the observable is observed (ie set to True) and set to 0 if the observable is not detected.
We define two distance functions :

- Equivalence : the distance is 1 if the points are different (time excluded), 0 otherwise.
- Hamming : the Hamming distance corresponds to the number of different values when comparing two points. We note that this distance corresponds to the edit distance, as presented in the previous section.

We are using a K-Means algorithm (Ferreira et al., 2012) in order to find clusters. The K-Means algorithm consists of the following steps :

1. (Initialisation) Choose k sequences S1, S2… Sk from the corpus. These sequences, noted C1, C2, … Ck are called *centroïds*.
2. For each sequence S from the corpus, compute the DTW algorithm between S and each centroïd. Each sequence is then associated with the closest (i.e. the minimum DTW computed value) centroïd.
3. For each centroïd C, compute the mean of each sequence associated with it. The computed mean becomes the new centroïd C.
4. Repeat 2 and 3 until all centroïds are stable (do not change during step 3) or until a determined number of iterations.
5. (End) The k centroïds are centroïds of k sets of sequences that are the clusters.

Step 3 requires a way to compute the mean of a set of sequences. For the CreaCube corpus, defining the mean of a set of sequences requires a way to compute the mean of multiple trajectories in a multidimensional boolean space. This research report focuses on one particular method called Dynamic Time Warping Barycenter Averaging (Petitjean et al., 2011).

2.2. DTW Barycenter Averaging

DBA stands for DTW Barycenter Averaging. It is an averaging method which consists in iteratively refining an average sequence in order to minimize the squared distance of the DTW value to average sequences.
Using the DTW algorithm to compare each sequence with the average, it is possible to associate a set of points from sequences to each point of the average, and then computing the barycenter of all points from sequences associated with each point from the average, thus creating a new average trajectory.

Let's consider the barycenter function :

$$Barycenter(\{X_1, X_2, ..., X_l\}) = \frac{X_1 + X_2 + ... + X_l}{l}$$

If we consider each point from the set as a n-dimensional vector, the addition sign corresponds to vector addition.
Given a set S, the sequence i of length m in the set is a succession of points :

$$S_i = s_{i1} s_{i2} ... s_{im}$$

where each point is a vector in a n-dimensional space :

$$s_{ij} = (s_{ij1}, s_{ij2}, ..., s_{ijn})$$

The average sequence A of length m' is written :

$$A = a_1 a_2 ... a_{m'}$$

where

$$a_j = (a_{j1} a_{j2} ... a_{jn})$$

For each point of A, we create a set of all points from sequences of S which are associated when applying DTW between A and sequences from S. The created space is :

$$Assoc(a_j)$$

The DBA algorithm averages each point of A individually by replacing it by the barycenter of all its associated points. In other words, at each iteration of the DBA algorithm, we do the following transformation for each point a of A :

$$a_j = Barycenter(Assoc(a_j))$$

The algorithmic method then consists of the following steps :

1. (Initialisation) : Create an initial average sequence A. This sequence can be random, preprocessed or even a sequence from the corpus
2. For each sequence S from the corpus, compute the DTW algorithm between S and A. For each point a from A, create a set s(a) containing all points associated with a from all sequences from the DTW algorithm.
3. For each point a from the average sequence A, compute the barycenter of all points from the s(a) set, coordinate by coordinate. The resulting barycenter becomes the new point a, thus modifying all points from sequence A.
4. Repeat 2 and 3 until the average sequence A is stable (does not change during step 3) or until a determined number of iterations.
5. (End) A is the average sequence.

Figure 7 shows an execution of the DBA algorithm with 4 different iterations over two one-dimensional trajectories, adapted from (Petitjean et al., 2011). Each point of the average sequence is associated with DTW to multiple points from orange and blue sequences, and each associated point is then a point used in the update of the average point with the barycenter function.

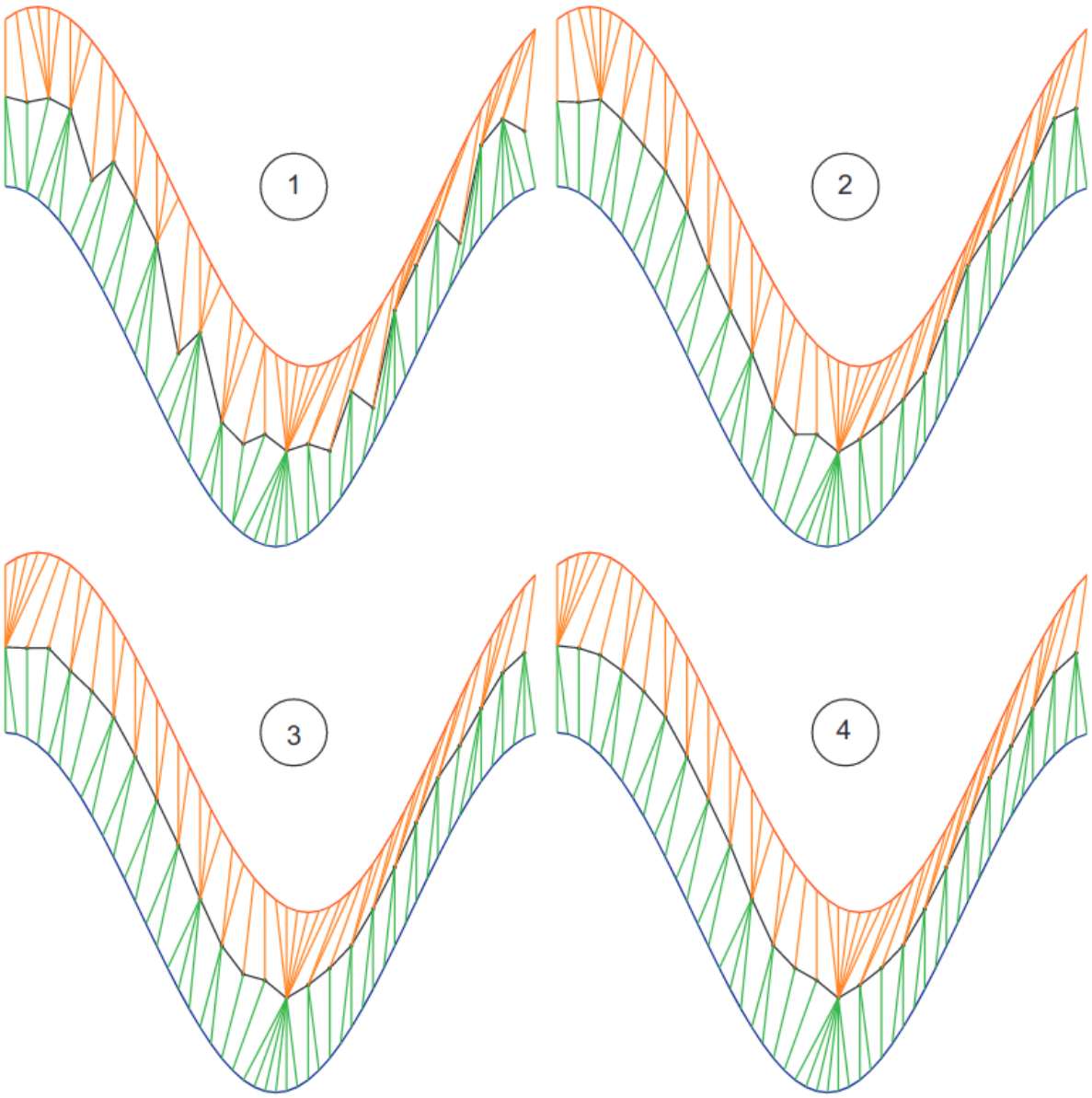

Figure 7 : An execution of the DBA algorithm over two-dimensional trajectories, from (Petitjean et al., 2011)

DBA is an iterative algorithm that allows to modify over time an average sequence, similar to the centroïds update of the K-means clustering algorithm. We need to adapt DBA to make the result correspond to what we are searching for, i.e. centroids.

2.3. Interpretation of DBA for centroid computing and boolean values

We will consider two different ways of considering DBA for centroïd computing : sequentially and concurrently
The first method is to consider using DBA as a way to compute the centroïd of a set of sequences : the computation of the centroïd consists of applying DBA with a maximum number of iteration and by giving a random sequence (or a sequence from the set) as the initial sequence to average.

The method, named *sequential DBA clustering*, will be the following :

1. (Initialisation) Choose k sequences S1, S2… Sk from the corpus. These sequences, noted C1, C2, … Ck are called *centroïds*.
2. For each sequence S from the corpus, compute the DTW algorithm between S and each centroïd. Each sequence is then associated with the closest (i.e. the minimum DTW computed value) centroïd.

3. For each centroïd C, compute the DBA algorithm with each sequence associated to C in step 2 with C as the initial sequence to average, until the averaged sequence is stable (does not change during the iteration step) or a maximum number of iterations is reached.
4. Repeat 2 and 3 until all centroïds are stable (do not change during step 3) or until a determined number of iterations.
5. (End) The k centroïds are centroïds of k sets of sequences that are the clusters.

The other method is to do each iteration of the K-means clustering algorithm and DBA concomitantly. For each iteration of the K-means algorithm, an iteration of DBA is done, and the iteration of DBA changes the value of the centroïd sequence.
This method, named *concomitant DBA clustering*, consists of the following :

1. (Initialisation) Choose k sequences S1, S2… Sk from the corpus. These sequences, noted C1, C2, … Ck are called *centroïds*.
2. For each sequence S from the corpus, compute the DTW algorithm between S and each centroïd. Each sequence is then associated with the closest (i.e. the minimum DTW computed value) centroïd.
3. For each centroïd C, compute the barycenter of each point from C with its associated set of points from sequences associated to C in step 2. The sequence consisting of the succession of each barycenter is the new centroïd C.
4. Repeat 2 and 3 until all centroïds are stable (do not change during step 3) or until a determined number of iterations.
5. (End) The k centroïds are centroïds of k sets of sequences that are the clusters.

Each method allows the computing of centroïds for the K-means algorithm. However, in the case of multidimensional vectors of booleans, and specifically for the study of the CreaCube task, multiple problems arise from the DBA execution :

Centroïd interpretability. As every variable from our corpus is a boolean value, the barycenter of boolean values is a value between 0 and 1. Those values are not interpretable unless, for instance, we consider a way to round the values to 0 and 1 to force boolean values. However, this is not possible for our corpus :
A sequence will have less than 15 values put to 1 over time, and every boolean value is considered True only at the moment the observable is detected. For instance, the discovery of an affordance is annotated as True only the moment the affordance is discovered. This leads to mostly “empty” sequences that, when doing barycenter averaging, will lead to empty sequences. In any case, resulting centroïds are difficult to interpret.
Explainability and plausibility If we consider that we found a way to avoid the “emptiness” of averaged sequences (by, let’s say, putting a threshold for rounding instead of a half rounding), we have another problem of explainability. Resulting centroïds have no guarantee to be plausible, i.e. be a sequence that could potentially be done by a subject. This is, among other things, due to the fact that our boolean values are not unrelated to each other. For instance, it is impossible for two different final structures (i.e. two structures composed of four cubes, but arranged differently) to be set to True at the same time.
Hierarchical boolean values Our observables dataset has related boolean values, that are for instance mutually exclusive. In the case of the computing of centroïds, these kinds of properties are not necessarily maintained.

For these reasons, and in particular the problem of explainability of centroïds, we do not think that centroïds are a good way to study behaviors among sequences. The use of DBA does not lead to satisfying prototypes that we can use to try to find properties of clusters.
We will prefer other algorithms to create more explainable clusters and prototypes (representative of a cluster).

Conclusion

DTW Barycenter Averaging is a method used to create an average sequence from a set of multi-dimensional sequences. However, the use of the barycenter function to compute the average makes it difficult to apply for boolean values. This leads to the creation of average sequences that are difficult to explain (because they are for instance not plausible) and to associate to behavior, which is the main point of our research. In addition, our corpus of boolean values has an implied hierarchical structure, which implies that some dimensions are more related to others (in particular booleans from the same category, see Figure 2). Using DBA requires at least the

definition of a distance that takes into account this hierarchical structure, which is not the case for distances we defined until now.
The definition of centroïds is a main point of interest for our research, because these can be considered as “typical sequences of actions” done by a subject. However, it appears that centroïds are not good candidates to represent this kind of typical sequence. Instead of centroïds, we will consider medoïds, which are sequences *from the corpus* : A medoïd of a group is the member of the group that is the closest to any other member of the group. We can compute it by applying the K-medoïds clustering algorithm, which is similar to the K-means clustering, with the advantage of producing real sequences from the corpus which are by definition plausible (because they exist within the corpus). The K-medoïd algorithm is as follows :

1. (Initialisation) Choose k sequences S1, S2… Sk from the corpus. These sequences, noted M1, M2, … Mk are called *medoïds*.
2. For each sequence S from the corpus, compute the DTW algorithm between S and each medoïd. Each sequence is then associated with the closest (i.e. the minimum DTW computed value) medoïd.
3. For each group, compute the mean square distance (with DTW) of each member with every other member. The member which has the smallest mean square distance is the new *medoïd* of the group
4. Repeat 2 and 3 until all medoïds and groups are stable (do not change during steps 2 and 3) or until a determined number of iterations.
5. (End) The k medoïds are medoïds of k sets of sequences that are the clusters.

This method corrects most of the problems created by the K-means algorithm and DBA, and will be used for future research as a way to find clusters and to find prototypes of such clusters used for comparison and interpretation. To correct remaining problems such as the hierarchical structure of our boolean values, future work will concentrate on the definition of numerical values evolving over time that could represent our boolean dataset.

Acknowledgement: This work is a collaboration between the ANR CreaMaker[5] and the AEx AIDE[6]
Thierry Viéville, Margarida Romero, Chloe Mercier and Frédéric Alexandre are gratefully acknowledged for precious advice during the execution of this work and this report reviewing.

Bibliography

Fansher, M., Shah, P., & Hélie, S. (2022). The effect of mode of presentation on Tower of Hanoi problem solving. *Cognition*, *224*, 105041. https://doi.org/10.1016/j.cognition.2022.105041

Ferreira, N., Klosowski, J. T., Scheidegger, C., & Silva, C. (2012). *Vector Field k-Means: Clustering Trajectories by Fitting Multiple Vector Fields* (arXiv:1208.5801). arXiv. https://doi.org/10.48550/arXiv.1208.5801

Glushkov, V. M. (1961). THE ABSTRACT THEORY OF AUTOMATA. *Russian Mathematical Surveys*, *16*(5), 1. https://doi.org/10.1070/RM1961v016n05ABEH004112

Knoblock, C. A. (1990). *Abstracting the tower of hanoi*. *4976*, 1–11.

Mercier, C. (2022). An Ontology to Formalize a Creative Problem Solving Activity. *IEEE Transactions on Cognitive and Developmental Systems*, 1–1. https://doi.org/10.1109/TCDS.2022.3210234

Molnár, G., Greiff, S., & Csapó, B. (2013). Inductive reasoning, domain specific and complex problem solving: Relations and development. *Thinking Skills and Creativity*, *9*, 35–45.

Müller, M. (Ed.). (2007). Dynamic Time Warping. In *Information Retrieval for Music and Motion* (pp. 69–84).

[5] https://creamaker.wordpress.com
[6] https://team.inria.fr/mnemosyne/en/aide/

Springer. https://doi.org/10.1007/978-3-540-74048-3_4

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Prentice-Hall.

Ney, H., & Ortmanns, S. (1999). Dynamic programming search for continuous speech recognition. *IEEE Signal Processing Magazine*, *16*(5), 64–83. https://doi.org/10.1109/79.790984

Petitjean, F., Ketterlin, A., & Gançarski, P. (2011). A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition*, *44*(3), 678–693. https://doi.org/10.1016/j.patcog.2010.09.013

Romero, M., Lille, B., & Patiño, A. (Eds.). (2017). *Usages créatifs du numérique pour l'apprentissage au XXIe siècle* (1st ed.). Presses de l'Université du Québec. https://doi.org/10.2307/j.ctt1vw0rkx

Tavenard, R. (2021). *An introduction to Dynamic TIme Warping*. https://rtavenar.github.io/blog/dtw.html

RESEARCH CENTRE
BORDEAUX - SUD-OUEST

351 Cours de la Libération
Bâtiment A29
33405 Talence Cedex France

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr
ISSN 0249-6399